USING PREDICTIVE ANALYTICS TO DETECT MAJOR PROBLEMS IN DEPARTMENT OF DEFENSE ACQUISITION PROGRAMS

THESIS

Austin W. Dowling

First Lieutenant, USAF

AFIT/GCA/ENC/12-03

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCA/ENC/12-03

USING PREDICTIVE ANALYTICS TO DETECT MAJOR PROBLEMS IN DEPARTMENT OF DEFENSE ACQUISITION PROGRAMS

THESIS

Presented to the Faculty

Department of Statistics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cost Analysis

Austin W. Dowling

First Lieutenant, USAF

March 2012

AFIT/GCA/ENC/12-03

USING PREDICTIVE ANALYTICS TO DETECT MAJOR PROBLEMS IN DEPARTMENT OF DEFENSE ACQUISITION PROGRAMS

Austin W. Dowling
First Lieutenant, USAF

Approved:

_____	_____
Edward D. White, Ph.D (Chairman)	Date
_____	_____
Jonathan D. Ritschel, Maj, USAF (Member)	Date
_____	_____
C. Grant Keaton, 1 Lt, USAF (Member)	Date

Abstract

This research provides program analysts and Department of Defense (DoD) leadership with an approach to identify problems in real-time for acquisition contracts. Specifically, we develop optimization algorithms to detect unusual changes in acquisition programs' Earned Value data streams. The research is focused on three questions. First, can we predict the contractor provided estimate at complete (EAC)? Second, can we use those predictions to develop an algorithm to determine if a problem will occur in an acquisition program or sub-program? Lastly, can we provide the probability of a problem occurring within a given timeframe? We find three of our models establish statistical significance predicting the EAC. Our four-month model predicts the EAC, on average, within 3.1 percent and our five and six-month models predict the EAC within 3.7 and 4.1 percent. The four-month model proves to present the best predictions for determining the probability of a problem. Our algorithms identify 70% percent of the problems within our dataset, while more than doubling the probability of a problem occurrence compared to current tools in the cost community. Though program managers can use this information to aid analysis, the information we provide should serve as a tool and not a replacement for in-depth analysis of their programs.

To my wife and daughter for their support and patience during this process, and for listening to me while I talk endlessly about how much I love Excel.

Acknowledgments

I would like to thank Dr. White, my thesis advisor, for providing me the tools and the guidance to pursue an interesting area of research. Dr. White enabled me to reach my full potential by guiding me and allowing me to pursue methods and research questions that interest me. I would like to thank the other members of my committee, Major Ritschel and First Lieutenant Grant Keaton, for providing their unique insight into my research.

Austin W. Dowling

Table of Contents

Page

Table of Figures

Page

List of Tables

Page

USING PREDICTIVE ANALYTICS TO DETECT MAJOR PROBLEMS IN DEPARTMENT OF DEFENSE ACQUISITION PROGRAMS

I: Introduction

DoD acquisitions demands qualified personnel to perform cost estimating and to track program performance. To maintain the current standard of DoD acquisitions, the acquisition's community must create new ways to complete the same task with fewer resources. In 2005, in an effort to cut costs, the Air Force reduced cost estimating personnel force to its lowest levels ever (Morin, 2010). In response to these levels, the Air Force Acquisition Improvement Plan sets out to re-affirm the acquisition management for the Air Force. The Air Force is currently rebuilding its acquisition force; however, in the meantime, the workload for the acquisition force exceeds the capabilities (Morin, 2010). The automation of tasks reduces the workload while still maintaining the performance that the field demands. Automating problem detection increases decision maker's awareness and decreases the likelihood of a program experiencing a cost overrun.

A prolific academic in the field of Earned Value Management (EVM) who has written over 20 articles, David Christensen (1992), shows that once a program exceeds the 20 percent completion point it cannot recover from a cost overrun. Early problem detection enables a manager to prevent these overruns and increase the stability of their program. Christensen (1992) also demonstrates in his research that if a contract portrays stability at the 50 percent completion point, it will remain stable until completion. The DoD uses Earned Value predictions to track their programs and prevent program instability.

Analysts currently use EVM to monitor performance of an acquisition contract. This analysis requires a large amount of time and a great understanding of EVM to determine the state of the contract. Analysts use various measures and ratios to develop their own estimates of future program costs. The analysts then compare these estimates to the estimates provided by the contractor to establish whether a problem might occur in their program (Headquarters Air Force Material Command, Financial Management, 1994). This comparison provides EVM analysts with an understanding of the overall direction of their program.

Using EVM data to determine the quality of a program is not a new idea. Analysts currently use various Earned Value techniques to evaluate their programs. Most analysts use ratios or charting techniques to assess trends in their programs. We address specific EVM further in our Literature Review Chapter. Keaton (2011) first addressed the use of an automated algorithm to evaluate a program. His algorithm compares various Earned Value ratios and relates changes in those ratios over time to significant changes in the estimate at complete (EAC). Keaton (2011) shows that an automated Earned Value management tool can detect future problems in acquisition programs; however, he did not provide significant insight to the relationships between various Earned Value data. We provide further detail regarding Keaton's methods in the next chapter.

Our Contribution

Analysts must synthesize all relevant information to ensure they provide decision makers with accurate and relevant information. To present decision makers with the best information, analysts need to understand the relationships that exist within the data. Our research not only provides the appropriate relationships to warrant the best information, but we also provide a methodology to determine those relationships within data.

Our data-mining algorithm, which we use for determining Earned Value relationships, can be applied to any data set. The methodology and procedures of our algorithm serves as a unique way to determine relationships and generate an accurate prediction model. Our algorithm not only selects the best variables to use in a model, it also adjusts the variables themselves to make them as predictive as possible. We address the data-mining algorithm in further detail in our Methodology Chapter. The outputs and findings of our algorithm prove significant for analysts and decision makers.

The results of this research provide decision makers with a tool to forecast the EAC measure up to six months into the future. We use those estimates to determine when, and with what likelihood, a problem will occur in an acquisition contract. Our research builds upon the original research established by Keaton (2011) and improves accuracy and the breadth of the research. The findings we provide facilitate a decision maker's understanding of the programs' status under his or her control. This increase in information allows program managers the needed oversight to correct instability issues before their programs reach 20 percent completion. Our research will not replace in-depth analysis that the field requires; however, we feel that our results will decrease the amount of oversight required to ensure a successful program.

Our research answers the following questions:

1. Can we provide an accurate point estimate for future contractor provided EAC's?
2. Can we detect future major changes to the EAC?
3. If we detect major changes to the EAC, can we provide decision makers with a timeframe and probability of those major changes to the EAC?

Chapter two, Literature Review Chapter, provides a brief overview of the current state of EVM and how the DoD acquisitions community uses it to monitor programs. The Literature Review Chapter presents a background of the previous research done by Keaton (2011). The chapter finishes with a background on the tools we use to provide predictions of the EAC. The Methodology Chapter reviews our method for determining our EAC predictions, how we detect problems, and how we use these detections to determine the probability of a problem occurring. In the Results Chapter, we present our findings. The Conclusions Chapter reviews our results and discusses the implications of our findings to the Department of Defense (DoD) and presents ideas for future research.

II: Literature Review

Cost growth plagues the DoD and leads to major budget problems. Analysts usually measure cost growth as a ratio of an early final cost estimate to the current estimate or the actual final cost of a program (Arena, Leonard, Murray, & Younossi, 2006). These estimates influence the decisions program managers make throughout the course of their project. Managers use the initial estimate to formulate a budget; therefore, if the program goes over the estimate, it exceeds the budget as well. Cost growth, as previously defined, proves rampant within the Air Force. RAND (2006) analyzed 220 completed weapon system programs from 1968 to 2003 and found a 46 percent average cost growth among all the programs analyzed. They also found that the longer duration programs had greater cost growth (Arena, Leonard, Murray, & Younossi, 2006). For example, the Spaced Based Infrared System currently exceeds the initial budget estimate by over 160% (Younossi & et al., 2007). Project management seeks to prevent cost growth or provide insight to future project changes.

In the late 1950's and early 1960's almost the entire aerospace and defense industries used project management (Kerzner, 2009). The DoD and construction companies started the use of project management techniques to enable them to track the status of their program (Kerzner, 2009). Managers use a variety of techniques to manage projects such as critical path analysis, risk monitoring and control, precedence networks, graphical evaluation and review technique, and many others (Kerzner 2009). EVM, another project management technique, gives project managers the ability to evaluate the status of their programs. The DoD uses EVM to track cost, schedule and technical performance of a contract. EVM uses ratios and different methods to predict the final cost of a program as well as track the status of the program. The Format-1 of

the CPR contains all the top-level EVM data we use to evaluate programs. For example, the Format-1 provides the Actual Cost of Work Performed (ACWP), broken out by work breakdown structure (WBS) level as well as all levels combined for the whole program. In our analysis, we use the combined levels that the Format-1 provides. Reference Appendix A for an example Format-1.

Earned Value expresses the amount of work done and the work remaining in monetary terms. In essence, EVM expresses a project's completeness in terms of cost or time (Erdogmus 2010). According to Bosch and Küttler (2011), practitioners of EVM, "The motivation for introducing EVM arises because project tracking often separates schedule monitoring from cost analysis" (Bosch & Küttler, 2011). Bosch & Küttlers' EVM knowledge derives from the Wendelstein 7-X project. They implemented EVM tools to monitor the Wendelstein 7-X project, a nuclear fusion reactor. The two found it difficult to establish a baseline schedule and break that schedule into definable packages. Additionally, they found it difficult when technical changes arose in the project. Overall, the two found EVM extremely versatile; although, they noted that managers must accompany EVM with other monitoring tools (Bosch & Küttler, 2011). This example is in line with the governments beliefs about EVM as provided in The *Guide to Analysis of Contractor Cost Data*.

The government requires the use of a DoD established system (the Cost/Schedule Control System Criteria-compliant management system) for the following: procurement contracts, modifications in excess of $250 million, or the test and evaluation phase in excess of $60 million. This system indicates work progress; relate cost, schedule, and performance; provide valid, timely, and auditable data; and provide a summarization of the information (Headquarters

Air Force Material Command, Financial Management, 1994). The system provides the information for the EVM analysis currently done in the acquisition community.

The *Guide to Analysis of Contractor Cost Data* provides an acquisition analyst with the necessary tools to evaluate CPRs and it acts as a manual for them to asses programs. The manual describes the data, which the contractor provides via the Format-1 of the CPR. Analysts use the cumulative ACWP, cumulative budgeted cost for work performed (BCWP), cumulative budgeted cost of work scheduled (BCWS), and the EAC to asses a programs performance (Headquarters Air Force Material Command, Financial Management, 1994). The manual presents the ratios, using the data from the Format-1, to evaluate cost and schedule reporting: schedule performance index (SPI), cost performance index (CPI), to complete performance index (TCPI), percent complete, percent spent, percent scheduled, and others (Headquarters Air Force Material Command, Financial Management, 1994). We address the use of these ratios further in the Methodology Chapter.

The *Guide to Analysis of Contractor Cost Data* further discusses ways for an analyst to determine if a problem exists in a program. According to the manual,

> Thresholds are established requiring a variance analysis for any cost or schedule variance that exceeds a certain percentage of BCWS or BCWP and/or exceeds an established dollar minimum….When initially establishing the thresholds, it may be advisable to provide for tightening these thresholds as the contract progresses" (Headquarters Air Force Material Command, Financial Management, 1994).

The manual also describes ways to forecast changes to the EAC as well as ways to use ratios, such as CPI and SPI, to determine possible problems. It recommends charting the ACWP, BCWS, and BCWP as well as the ratios, previously addressed, over the time of the project to visually display and analyze the changes. By doing this, the analyst determines the overall performance of a project. The government requires the program manager to define significant

variances and thresholds to determine when a program has a problem (Headquarters Air Force Material Command, Financial Management, 1994). The manual does not tell analysts which method of EAC calculation provides the most accurate results, nor does it provide a numerical way to forecast a problem in the program. The manual does not present a way to use changes in ratios or monthly data; rather, it only uses point estimates or three month averages to forecast the EAC. We list all the ratios and variables the manual references in Appendix B. Keaton (2011) addresses this concern with his time series analysis.

Previous Research

Keaton (2011) analyzed the CPI and SPI with time series Autoregressive/Integrated/ Moving Average (ARIMA) models. He showed that an analyst could model the CPI and SPI through a first difference model (Keaton et al., 2011). Using a control chart to monitor the CPI and SPI, he detected potential problems in a program, which therefore created different bounds of the control chart. He defines a problem as an absolute change in the EAC greater than five percent from one month to the next. When a reported CPI or SPI fell out of the expected range, his algorithm demonstrated a time-lagged relationship to future problems.

He looked at different standard deviations for the bounds of the control chart, from 0.5 standard deviations to 3 standard deviations, where the standard deviation updates with new information. He found that the higher the standard deviation the less likely a false positive, but the greater likelihood for a missed detection. In addition, he found no relationship between consecutive detections and the likelihood of a significant change in the EAC (Keaton et al., 2011). His algorithm does not provide analysts with the information of when or with what probability a problem will occur. In addition, his algorithm does not forecast the magnitude or direction of the change in the EAC, only that a change of greater than five percent will happen

within a year of the detection. We use his findings and take them a step further by forecasting the EAC.

Many different industries use forecasting and time series analysis to gain insight into future events. Analysts classify forecasting problems by time: short-term, medium-term, and long-term. Short-term forecasts sometimes only span a few days while log-term forecasts can extend beyond a few years. To generate forecasts, researchers use past data to generate statistical models to predict a future event. These forecasts usually influence the strategic planning of the various fields. When analysts try to predict too far beyond the scope of the data, poor forecasts ensue. For example, in 1966 the *Wall Street Journal* predicted, "Computers are multiplying at a rapid rate. By the turn of the century there will be 220,000 in the U.S" (Montgomery, Jennings, & Kulahci 2008). In actuality, 54 million households possessed at least one computer representing over half of all households (U.S Department of Commerce 2001). To provide useable forecasts to decision makers, analysts need to possess at least background of basic forecasting principles.

Forecasting

Quantitative forecasting enables researchers to anticipate future outcomes and apply probabilities to future events (Makridakis, Wheelwright, & Hyndman, 1998). Decision makers only need to use forecasts for uncertain and uncontrollable events (Armstrong, 2001). Researchers constantly work to improve forecasting techniques and errors in forecasting decrease as a result. For example, before 1987, analysts predicted 27 percent of tornados compared to 59 percent by 1997 (Armstrong, 2001). All forecasting models follow a universal form of Equation 2.1:

$$observation = pattern + error \tag{2.1}$$

Quantitative forecasting relies on two principals. First, the past events must be quantifiable. Second, the researcher expects the pattern to repeat in the future or the data presents evidence that the pattern repeats (Makridakis, Wheelwright, & Hyndman, 1998).

Armstrong (2001) presents four principles to follow in his book *Principles of Forecasting:*

1. Use all the data possible.
2. When developing quantitative models, researchers must make the models simple.
3. Do not use personal judgment to revise predictions from forecasting models.
4. Researchers should investigate theory prior to developing quantitative models.

When analysts do not follow these principles, their models can produce poor predictions. For example, prior to the energy crisis of 1970, researchers did not use all the available data to develop their models and the model produced results, which led to the energy crisis (Armstrong, 2001). In addition, no forecasting at all leads to uninformed decisions; therefore, it proves essential to provide decision makers with reliable insight to future events. Researchers use many different methods to forecast events; in our research, we use linear regression.

Linear regression, commonly used as a mathematical forecasting technique, is one of the widely used and most common forecasting techniques. It provides a way of relating various attributes, which act in a predictable manner, to a response or outcome. Linear regression uses explanatory variables to forecast a response variable. Time series analysis, through linear regression, uses previous responses to predict future response (Shumway & Shumway, 2000). Equation 2.2 represents a general form of linear regression equation (Gross, 2003).

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon \tag{2.2}$$

In Equation (2.2), "Y" represents the response variable with a given time unit and "β_p" represents the coefficient of the explanatory variable. The coefficient portrays the average effect on the response per unit increase in the "X" variable associated with the respective coefficient. To compare parameter estimates and establish which explanatory variables have the most impact on the response variable, analysts typically use standardized coefficients.

Standardized Coefficients

Standardized coefficients represent the relative impact of the explanatory variable on the model. Standardizing the variables requires that all variables portray a value of one standard deviation, which enables an even comparison between variables. Equation 2.3 demonstrates how to standardize a variable.

$$y_{standardized} = \frac{y_i - \bar{y}}{s} \tag{2.3}$$

In Equation 2.3, "y_i" represents the individual value for the variable, "$\bar{y}$" represents the average of all "y" variables, and "s" portrays the standard deviation within the variable. This equation turns the variables, used in the "X" matrix of a regression, into variables with the same scale. This allows for an equal comparison between the variables, which enables analysts to determine which variables influence the model the most. (Wiley, 2002). We use standardized coefficients in our stepwise regression algorithm. The standardized coefficients serve as a way to establish variables to remove; we go into further detail about how we remove variables from our stepwise regression in our Methodology Chapter.

Stepwise Regression

Three types of stepwise regression exist: forward, backward and mixed stepwise regression. In forward regression, the algorithm starts with no variables, and then adds one of

the variables to the model. If the variable improves the model, it stays in the model. Contrarily, if the variable fails to improve the model, the algorithm does not include it. All three stepwise regressions use a t-test to determine significance. For a forward regression, the analyst sets the significance levels to determine if a variable improves the model (Bart, Flinger, & Notz, 1999). This process repeats until the algorithm tests every variable. Backward stepwise regression works the opposite of forward regression. The backward stepwise regression enters all the variables into the model and removes the variable with the greatest p-value until all the variables in the model meet the analyst's p-value requirements. The algorithm repeats this process until all the variables' p-values meet the minimum cut-off p-value. In mixed, also referred to as full stepwise, the stepwise algorithm alternates between adding and removing variables. The mixed stepwise algorithm will add in variables removed earlier and test their significance (Bart, Flinger, & Notz, 1999). Each of the three stepwise techniques presents different advantages and disadvantages.

Backward stepwise regression presents advantages over forward regression, "backward-deletion variations is often preferable to the forward-selection variation because of its ability to deal with suppressor effects, which occur when a predictor has a significant effect but only when another variable is held constant" (Andrew, Pedersen, & McEvory, 2011). Backward stepwise regression requires more computational power than forward regression but less than the mixed regression (Bart, Flinger, & Notz, 1999). When analysts require more exploratory research, stepwise regression can determine the significance of new relationships (Andrew, Pedersen, & McEvory, 2011). For our analysis, we use time series data within our own backward stepwise regression.

Time Series Analysis

An analyst must first perform preliminary work prior to making a time series model. The analyst must think about the following prior to model building:

1. Ask the right questions to get background information
2. Determine clear objectives to produce the forecast
3. Establish exactly how the forecast will be used
4. What variables should be included/excluded (Chatfield, 2000)?

Analysts must also avoid unfairly improving a model by:

1. Using the validation data while making the model
2. Fitting multiple models to the test set and choosing the best results
3. Using variables that contain data from the time period of the prediction (Chatfield, 2000).

The success of time series models depend upon identifying the underlying trends and the relationships of the inputs (Peterson & Pi, 1994). The integration of Chatfield's (2000) procedures enables an analyst to study the dependency between the response variable and the prediction variables. To incorporate time series data into the linear regression in Equation 2.2 an analyst replaces the "X" variables with previous "Y" variables shown in Equation 2.4.

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \cdots + \beta_i Y_{(t-i)} + \varepsilon \tag{2.4}$$

The "β" parameters in Equation 2.3 represent the average impact on the predicted time period for the corresponding "Y_{t-i}". In a time series analysis, the predictive variables can take on different values other than previous "Y" values; however, all the variables must only include previous data. For example, in a time series model a researcher might use the standard deviation

of the "Y" values from four periods ago to the last period. Time series assumes a relationship exists between previous data and future data.

When researchers extrapolate beyond the data, they run the risk that their model will act differently in the future (Montgomery, Peck & Vining, 2011). Therefore, researchers must build the most accurate model to ensure viable future predictions. In order to build the most accurate model, a researcher must determine the global and local minimums of the error term. Cutting the Plane and the Simplex Method are two widely used linear programming techniques to solve complex problems and develop the most accurate models. Many businesses, in numerous fields, use linear programming to solve complex problems. A survey of Fortune 500 companies found that 85% of respondents use linear programming in their businesses (Harshbarger & Reynolds, 2008). These linear programming techniques improve the accuracy and the value of the forecasts.

Cutting the Plane

The Cutting the Plane method is a tool used to solve convex optimization problems. Convex optimization problems present themselves when no analytic solution exists. The surface of a convex optimization problem can take on both convex and concave or just a concave shape. Many different solutions exist in a convex optimization problem (Boyd & Vandenberghe, 2004). The cutting plane algorithm seeks to determine the global minimum or maximum. Different cutting plane methods exist. Ralph Gomory (1960), a recognized American mathematician, developed a method referred to as the fractional method. In the fractional method, the term an analyst optimizes changes through equal fractional cuts to determine the lowest or highest value, which the analyst optimizes. The fractional cutting method ensures that the analyst determines the true optimal solution; however, to ensure this convergence, the method requires a large

number of cuts (Batson, Chen, & Dang, 2010). Analysts use the Cutting the Plane method in such fields as integer programming and linear programming.

Analysts use these algorithms to solve two common optimization problems, the *traveling salesman* problem and the *linear ordering* problem (Floudas, 2001). According to Gutin and Punnen, "The traveling salesman problem is to find a shortest route of a traveling salesperson that starts at a home city, visits a prescribed set of other cities and returns to the starting city" (Gutin & Punnen, 2002). Mitchell and Borchers (1998) describe a real world example of the linear ordering problem.

> As an example of the aggregation of individual preferences, consider a tournament between a number of sports teams, where each team plays every other team. We wish to determine which team is the best, which is second best, and so on. If Team A beats Team B then Team A should finish ahead of Team B in the final ordering. However, it may be that Team B beat Team C, who in turn beat Team A. Therefore, it is not generally a simple matter to determine the final ordering. We could just count the number of victories of each team, but this may not truly represent the relative strength of some teams, and it may well lead to ties in the ordering. Therefore, we usually take the margin of victory into account when determining the final ordering (Mitchell & Borchers, 2000).

We go into further detail as to how we implement the Cutting the Plane method in our Methodology Chapter of this paper. After using a method for determining the approximate location of the global minimum or maximum, we use a modification of the Simplex Method to determine the local minimum. We assume the local minimum equals the global minimum since the local minimum resides near the approximate location of the global minimum, which we derive from the Cutting the Plane algorithm.

Simplex Method

The Simplex Method is a tool for finding the local minimum or maximum. The method adapts itself to the local landscape in order to find the minimum or maximum. The method does not rely on derivatives or advanced math and is computationally compact. The method only

requires that the surface present a continuous function. (Mead & Nelder, 1965). Nelder and Mead define the function value, y_i, for the minimization of a function at $P_0, P_1 \ldots P_n$ points in n-dimensional space, which defines the "simplex". Equation 2.5 represents the general form of the equation:

$$P^* = (1+\propto)\bar{P} - \propto P_h \tag{2.5}$$

Nelder and Mead describe the Simplex process,

> Where "$\propto$" is a positive constant, the *reflection coefficient*. Thus $P^*$is on the line joining P_h and $\bar{P}$, on the far side of $\bar{P}$ from P_h with $[P^*\bar{P}]$. If Y^* lies between y_h and y_i, then $P_h$is replaced by P^* and we start again with the new simplex. If $y^* > y_i$, i.e. if reflection has produced a new minimum, then we expand P^* to P^{**} by the relation $P^{**} = \gamma P^* + (1-\gamma)\bar{P}$. The *expansion coefficient* γ, which is greater than unity, is the ratio of the distance $[P^{**}\bar{P}]$ to $[P\bar{P}]$. We then accept P^{**} for the P_h and restart (Mead & Nelder 1965).

After the algorithm finishes, the "P" value represents the local minimum for the function ("Y"). Harshbarger and Reynolds describe the method in simple terms, "This method gives a systematic way of moving from one feasible corner of the convex region to another in such a way that the value of the objective function increases until an optimal value is reached or it discovered that no solution exists" (Harshbarger & Reynolds, 2008). Analysts first used the Simplex Method when dealing with scheduling problems that arose from the 1948 Berlin airlift. The analyst maximized the amount of goods delivered with various constraints. Since then, analysts use the Simplex Method to solve many different optimization problems across a large variety of businesses (Harshbarger & Reynolds, 2008). We provide further detail for the use and simplification of the Simplex Method in the Methodology Chapter.

In the next chapter, we detail how we collect our data, its limitations, and its breakout. We explain how we separate our data, which enables us to validate our results. Subsequently, we review the procedures we use to determine the optimum models to predict the contractor

provided EAC. We provide an in-depth review of the process we use to establish our variables and the parameter estimates that go along with those variables. After describing our algorithm, we detail the steps we take to use our model outputs to generate probabilities of a problem occurrence. Finally, we conclude the chapter with our procedures for establishing the validity of our results.

III: Methodology

This chapter details our procedures for using Earned Value data to forecast potential problems in acquisition programs. We first describe our data set, its limitations, the measures we extract from it, and how we standardize the data prior to developing our model. Then, we explain our optimization techniques: Cutting the Plane, the modified Simplex Method, and Ordinary Least Squares (OLS), into our model building process to ensure we select the most predictive explanatory variables. We then discuss how we use our model outputs from our four to six-month predictive models to forecast the contractor EAC, and how we use those forecasts to generate a control chart that predicts the likelihood of a problem occurrence. We define a problem as an absolute five percent change in the EAC, the same as Keaton (2011). We finish this section with a review of how we validate our models and control chart.

Data Source

We obtain all our data from the Defense Cost and Resource Center (DCARC). This database stores the acquisition contract information for major acquisition programs. We use the CPRs provided by the contractor to obtain our data. These CPRs come in many formats: Portable Document Files (PDF), Hyper Text Markup Language (HTML), Excel, and Extensible Markup Language (XML). We only analyze at the PDF, HTML and Excel files. We do not use XML files because we do not possess the unique program the contractors use to create them and therefore cannot extract the data.

We initially search DCARC to obtain possible acquisition category 1D (ACAT ID) programs to collect. We limit ourselves to using ACAT ID programs because these programs contain the most oversight and cost the most money, which, in turn, cause the greatest

consequences when a major problem occurs. We use all the DCARC data except when the program contains less than 10 consecutive months of data or we encounter unreadable data. We find 37 unique usable programs or sub-programs containing 1304 months of data; Appendix C lists the data by program. Table 3.1 contains all possible DCARC data. The "All Programs" row of the table refers to the programs within DCARC that contain CPR data and the "ACAT ID" row refers to how many of the total programs are ACAT ID. The "useable" row represents how many programs contain enough data, in the right format, and do not contain major data gaps of the ACAT ID programs. Our data covers programs from all the services and spans different types of programs with dates ranging from September 2007 to August 2011. Refer to Table 3.2 for the breakout of programs by service type and Table 3.3 for a breakout by type of program.

Table 3.1: Data Available in DCARC

Category	Number of Programs
All Programs	118
ACAT ID	64
Useable	37

Table 3.2: Number of Programs by Service Type

Service	Number of Programs
AF	14
Navy	8
Army	7
Joint	7
Marine	1

Table 3.3: Number of Programs by Type of Program

Type	Number of Programs
Plane	10
Comm.	9
Satellite	5
Missile	3
Helicopter	3
Radar	2
Ship	2
Facility	2
Vehicle	1

To ensure the accuracy of our models, we create a validation set of data, before starting our analysis. We use a 20 percent stratified random sample from our original data set. We ensure that 20 percent of the data comes from "small" programs, less than 30 months of data, "medium" programs, between 30 and 40 months of data, and "large" programs, more than 40 months of data. For instance, if we have 10 small programs we ensure we use two of those programs for the validation set. We use Excel's® random number generator to choose which programs we use in our validation set. We use eight programs for validation containing 276 months of data. This represents 21.6 percent of the programs and 21.0 percent of the months of total data. Table 3.4 depicts the programs size and if we use them for analysis or validation. Reference Appendix C for a complete breakout of our validation set.

Table 3.4: Programs by Months of Data

	Small Programs (Less than 30 Months of Data)	Medium Programs (30-40 Months of Data)	Large Programs (More than 40 Months of Data)
Number of Programs (Analysis)	9	7	13
Number of Programs (Validation)	2	3	3

Data Limitations

Our data faces four unique limitations. An explanation of how we addressed each limitation is provided. We provide an explanation of how we address each limitation. Currently, DCARC is the only database that provides complete CPR Format-1 data. Therefore, our first limitation is that we only collect data from one source; however, DCARC compiles data from multiple contractors and multiple sources so it only appears that we have one source of data. For our second limitation, we come across one-month gaps within a program, where DCARC does not provide data. Ten one-month gaps exist in our data set, which accounts for less than one percent of the data. To address this limitation, we use a linear approximation to fill the hole in the data. For example, if DCARC does not provide CPR data for February 2010, but DCARC presents CPR data for January and March of 2010, to determine the value of February of 2010 we use the average of January and March 2010. For instance, if January presents an ACWP of 1000 dollars and March presents an ACWP of 1200 we use 1100 dollars for February's ACWP. We repeat this procedure for all the data we use in our algorithm, which we discuss in further detail later in this chapter. A linear approximation will reduce the variability. Major gaps in our data, larger than one month, present our third limitation; if DCARC does not provide more than one month of consecutive data, we stop analyzing the program. For example, if a program has consecutive data from January of 2008 to May 2010 and data from August 2010 to February

2011, we exclude the data from August 2010 to February 2011. Sometimes the data DCARC provides does not cover the entire program. Meaning DCARC might only provide the first 40 consecutive months of data when the actual program lasted 50 months. Our final limitation deals with the limited variance within our response value.

If we use less than a four-month prediction, the response values, the ratio of the EAC's (refer to Equation 3.1), do not provide enough variation to determine statistically sound parameter estimates. Meaning, our parameter estimates will depend on only a few months of data. Therefore, the data forces us to predict no less than four months out.

$$\frac{\text{EAC}_{4-6\text{ Months ahead}}}{\text{EAC}_{\text{Current Month}}} \tag{3.1}$$

For instance, if we only predict one month out, the data only presents the change for one datum; however, if we predict four months into the future, the data presents us with four opportunities to detect the pattern that relates the change. For example, if we predict four months into the future and a major change occurs in month nine, our response variable has months five through eight to detect that pattern relating to the change; contrarily, a one-month prediction would only have month eight to detect the pattern. Essentially, the more opportunities to predict a change in the EAC, the better the chance there is for us to determine the pattern within the data. In addition, our data contains 67 instances where the contractor provided EAC changes by greater than an absolute five percent, (our definition of a problem), from one month to the next. We lose 12 of the 67 instances because they occur in the beginning of program, which resides outside the prediction window of our models.

Response and Explanatory variables

As previously discussed, we obtain all of our data from DCARC for our analysis. We specifically use the Format-1 data from the CPRs that the contractor provides to DCARC. The

Format-1 data consists of Earned Value data for the current period, cumulative, and at complete values for each WBS element. Since we provide predictions and problem detections at the overall program level, we use top-level WBS data. Top-level refers to using the summation of all the different WBS levels for each component. For example, we sum all the different WBS BCWP components to get the top-level BCWP. From the Format-1, we collect the following earned value data: contractor provided EAC (best, worst, and most likely), cumulative BCWP, cumulative BCWS, and cumulative ACWP.

For our response variable (y), we use a ratio of the most likely EAC, reference Equation 3.1. Our model uses three different ratios of four, five, or six month out predictions. For our explanatory variables ($x's$), we use ratios; which we derive from the cumulative BCWP, cumulative BCWS, and cumulative ACWP. Refer to Appendix B for definitions if needed. These ratios, for both the response variable and the explanatory variables, standardize the variables between our different programs, which enables us to compare multiple programs at the same time.

We initially create 148 variables to consider, shown in Appendix D. To address the large magnitude of variables, we perform an initial screening to reduce the number of explanatory variables to a useable number. To reduce the number, we perform a regression analysis between the most likely EAC and the explanatory variables, only using those variables with a p-value less than 0.1 (our significance level). This procedure reduces the number of explanatory variables to 30 variables. Appendix E contains a breakout of our 30 variables and the equations we use to derive them. Later in this chapter, we go into more detail regarding our initial screening process along with how we arrive at our final explanatory variables for their respective models.

Overall Algorithm Flow

To provide a better understanding of our algorithm, reference Appendix F for the Visual Basic code, we outline its flow:

I. Initial Screening – Use a mixed stepwise regression to reduce the number of variables from 148 to 30. We perform this procedure one time prior to starting the algorithm.

II. Reduce the Number of Significant Variables from 30 to 12, including the intercept (to keep the model simple). The algorithm performs the steps within this procedure until the variables meet the significance level and quantity.

   a. Optimize the Variables

      i. Cutting the Plane (using OLS)

      ii. Modified Simplex (using OLS)

   b. Remove a Variable

      i. Determine if all the variables meet the required significance level.

         1. If the variables all meet the required significance, remove the variable with the least impact to the model.

         2. If the variables do not meet the required significance level, remove the variable with the least impact to the model and one that does not meet the significance level.

   c. Check Variable Quantity

      i. If all variables are significant and there are less than 12 variables, move to section III.

      ii. If there are more than 12 variables or some of the variables are not significant, go to section II.

III. Determine the Optimum Order to Optimize Threshold Variables

IV. Using the 12 or less Variables, Minimize the MAPE

V. Using Forecasts from Section IV, Generate Control Chart Bounds

VI. Validate

   a. Forecasts from section IV

   b. Predictions from Control Chart, section V

Variable Selection

We use a standard OLS model previously listed in the literature review section (Equation 2.2) to determine our variables and their thresholds. Before we begin our backward stepwise regression, we reduce the possible variables to 30 in order to make the data manageable. To reduce the variables we use a mixed stepwise regression, explained in the Literature Review Chapter, with an exclusion criterion p-value of 0.1 for the variables. After we prepare our variables for analysis, we obtain our parameter estimates, by minimizing the sum of squared error (SSE). We use both static and dynamic variables in our model. SPI presents a good example of a static variable since it does not change because the components that makeup the equation do not change, while dynamic variables in our model change based upon a given input. For example, one of our variables, Large CPI, presents a value of one if the CPI presents a value larger than some threshold and a zero otherwise, (reference Equation 3.2).

$$CPI\ Large = \begin{cases} 1\ if\ CPI > Threshold \\ 0\ otherwise \end{cases} \quad 3.2$$

Appendix E lists which of our variables use thresholds and which do not.

To determine thresholds in our analysis, we use the Cutting the Plane method and a variation of the Simplex Method, both previously described in the Literature Review Chapter. We use these two methods, in conjunction with one another to determine significance thresholds,

to optimize the function itself. We then remove variables to meet our conditions concerning variable significance and quantity. For our analysis, we ensure that the overall p-value displays a value less than 0.005, where the null hypothesis states that no difference exists between zero and the population parameter. We use a lower than generally acceptable, 0.05, value to ensure our variables significance, even if we fail to meet all the OLS assumptions. We limit the number of variables we use in our model to 12, including the intercept, or less to keep our model simple. We address this later in the section in the variable removal portion.

Once the model meets the significance and variable quantity conditions, we optimize the parameter estimates by minimizing the mean absolute percent error (MAPE). Optimizing with MAPE instead of SSE ensures even weighting of each individual month of data. Due to computational difficulties, we do not use the MAPE to determine variable thresholds and selection. To minimize MAPE, a non-linear function, we use Excel's SOLVER, which converges on the solution through maximum likelihood estimators (Rachev, 2007). Using the maximum likelihood estimators requires significantly greater processing power than minimizing the SSE, minimized through a linear process, which forces us to use the SSE as our loss function for determining variables and their thresholds.

Cutting the Plane

We use the Cutting the Plane method, previously described in the Literature Review Chapter, to determine the approximate location of the global minimum of the variable's function in which we optimize. For example, using "Large CPI," defined by Equation 3.2, we determine the approximate "threshold" that minimizes the SSE. To determine this threshold, we first apply a range of possible solutions for the function to ensure we do not overweight a few months of data. For example, the threshold for the CPI function takes on any number between one and

1.15. We then divide the range into 20 equal cuts and determine the SSE for each value. For instance, using the "Large CPI" range, we test 1, 1.0075, 1.015, 1.025…1.15 and determine the SSE for each threshold. Figure 3.1 displays a visual representation of the Cutting the Plane algorithm. We consider the value that displays the lowest SSE the approximate location to the global minimum for that variable and the starting point for the modified Simplex Method.

Simplex Method

After the algorithm determines the approximate global minimums, we use a variation of the Simplex Method to determine the local minimum. As we previously stated in the Literature Review Chapter, we assume the local minimum equals the global minimum because the starting point for the Simplex Method resides near the global minimum. In our version of the Simplex Method, we use the percent change in SSE to determine whether the algorithm continues or stops at the given solution.

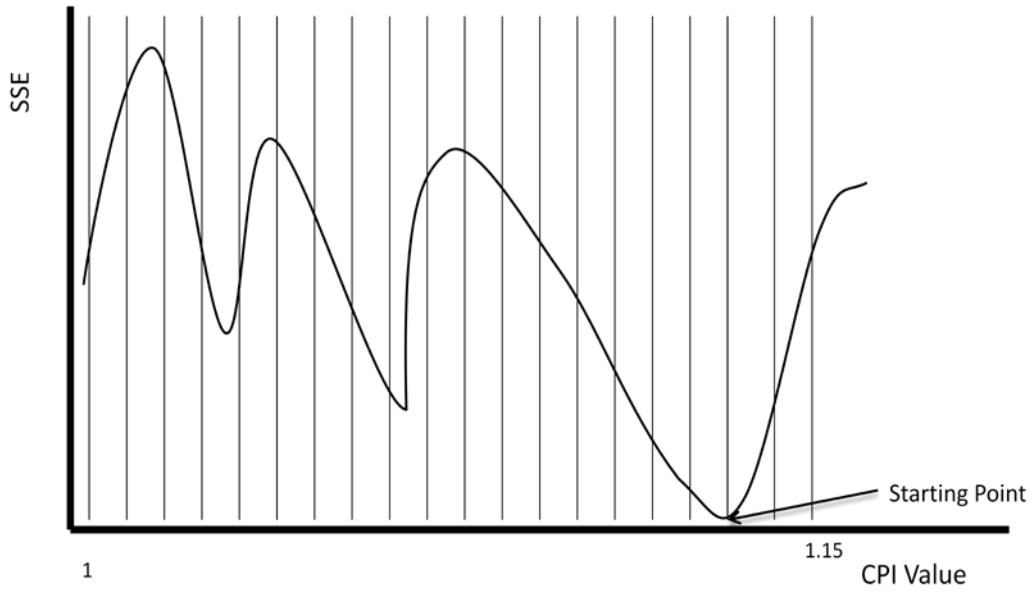


Figure 3.1: Cutting the Plane Example

Figures 3.2, 3.3, and 3.4 outline our process to determine the local minimum.

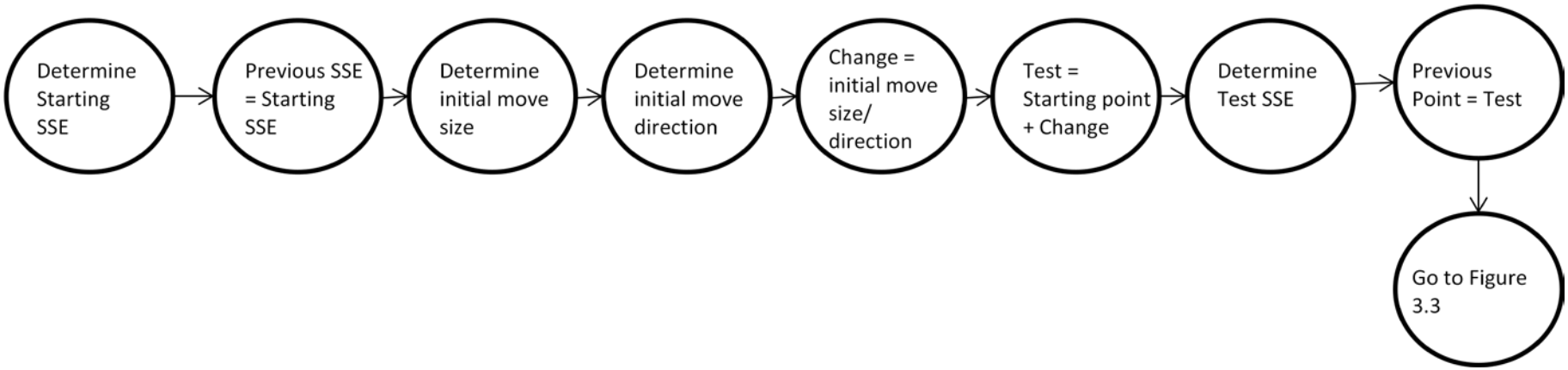


Figure 3.2: Modified Simplex Method Initial Procedure

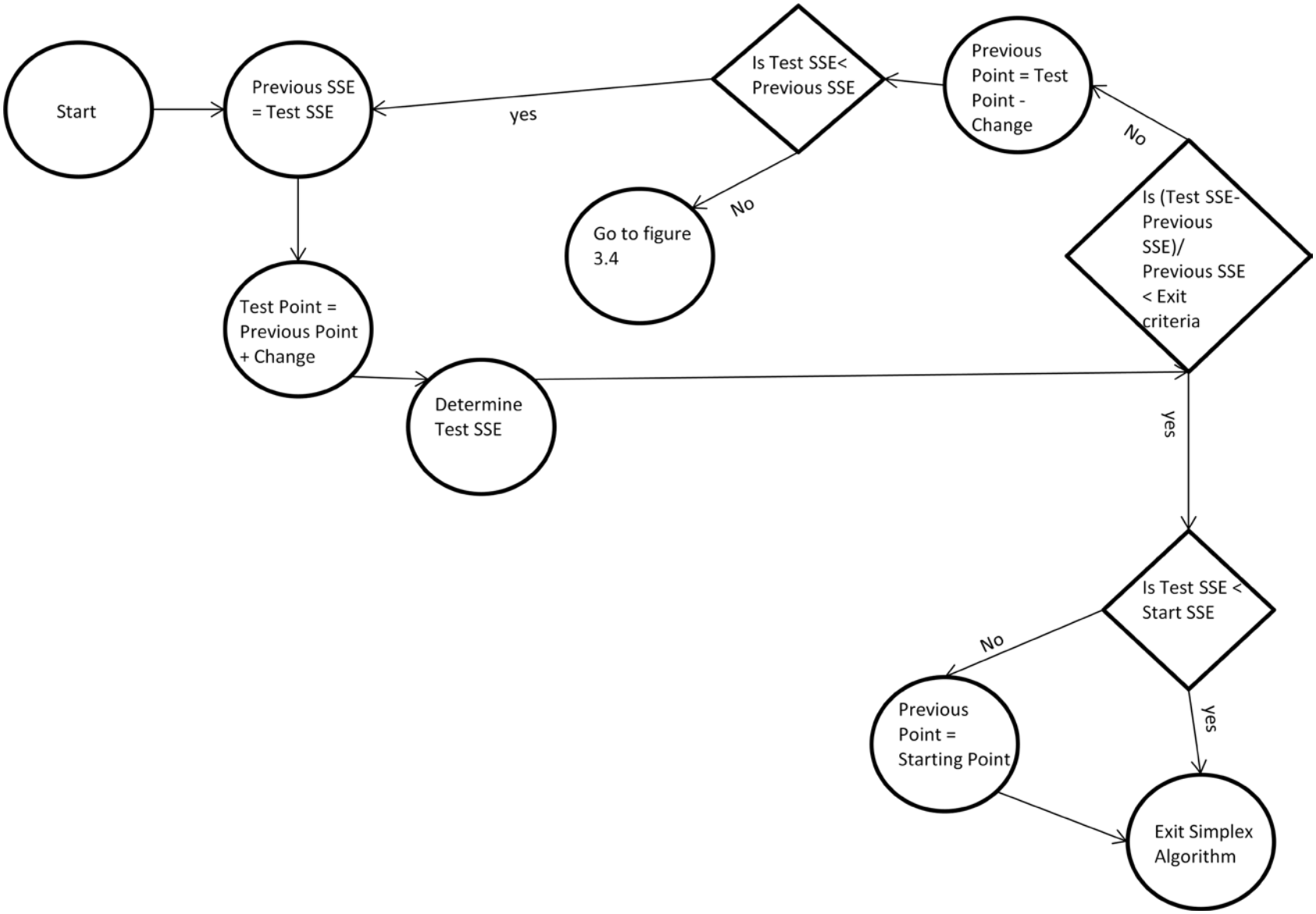


Figure 3.3: Modified Simplex Method Test SSE<Previous SSE

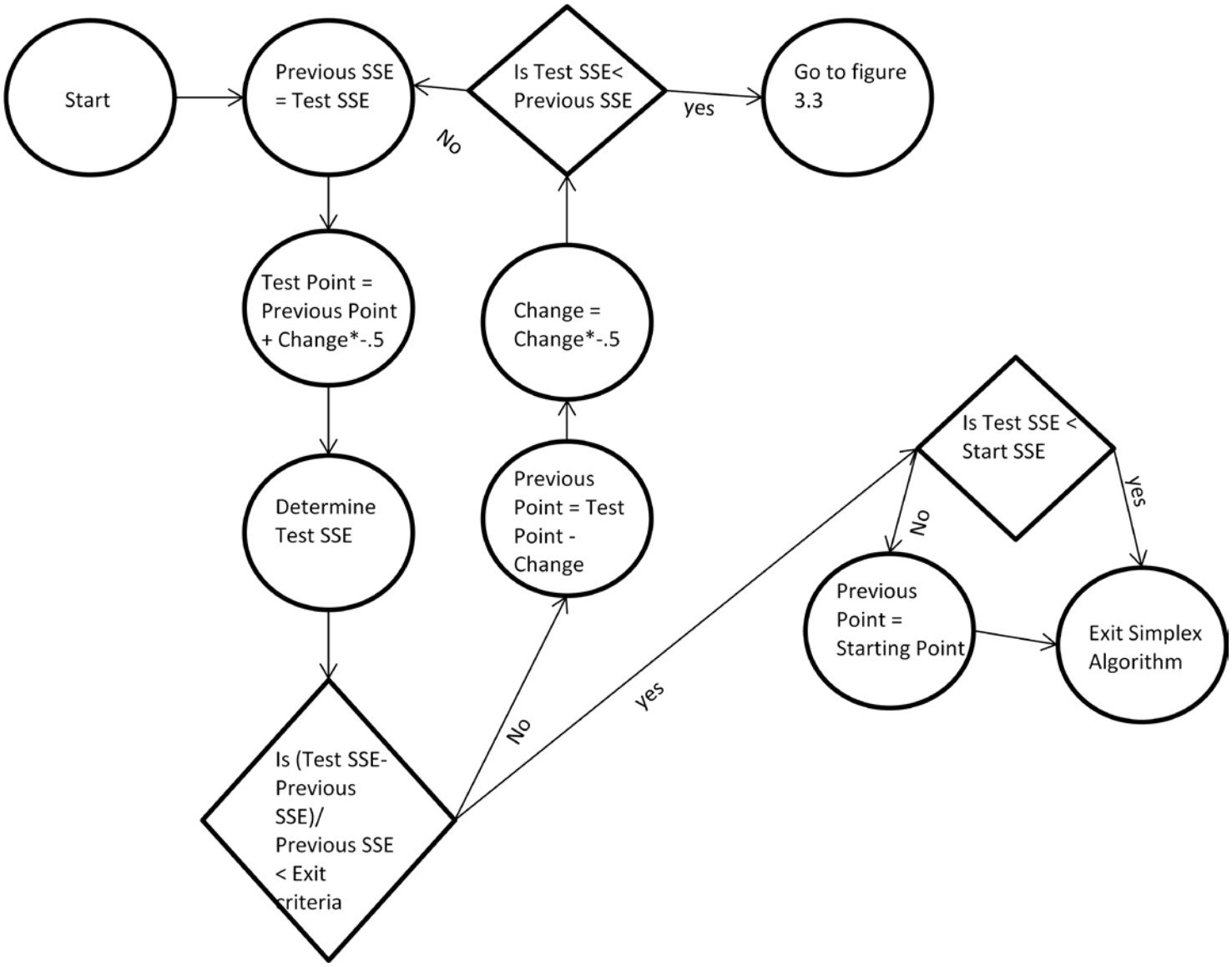


Figure 3.4: Modified Simplex Method Test SSE>Previous SSE

The starting SSE equals the lowest SSE from the Cutting the Plane algorithm. To determine the magnitude from the starting point, we divide the range, as described in the Cutting the Plane section, by 40. This represents half the distance between the cuts from the Cutting the Plane method. For example, on the "CPI Large" variable our range portrays a value of 0.15; therefore, 0.00375 represents half the value of the distance between the cuts. We determine if we increase or decrease the starting point by comparing the SSE of a positive change and negative change from the starting point. For example, if the lowest SSE, from the Cutting the Plane method, displays a value of 1.04, then 1.04375 and 1.03625 correspond to the two test points to determine the initial move direction. The algorithm will continue to change the previous point by the change, same magnitude and direction, until the previous point's SSE generates a lower SSE than the current point's SSE. For example, if 1.04375 displays a lower SSE than 1.03625 and the starting point (1.04), then the algorithm would then test 1.0479 (1.4375 + 0.00375). This process will continue until the SSE increases. When this happens, the change decreases in magnitude by half and changes in direction from positive to negative or negative to positive. For example, if 1.0479 portrays a SSE larger than 1.04375, then the algorithm will test 1.046025 ($1.0479-\frac{0.00375}{2}$). This process of changing magnitude and direction continues until the percent change in SSE exhibits a value less than the exit criterion.

For our algorithm, we use an exit criterion of $1 * 10^{-6}$. After the algorithm meets the exit criterion, it compares the starting SSE to the final test SSE. If the final test SSE portrays a value less than the starting SSE, then the final test point becomes the optimum threshold for the function the algorithm optimizes. However, if the starting SSE depicts a value less than the final test SSE, then the starting point becomes the optimum threshold for the function.

Variable Removal

After the algorithm runs the Cutting the Plane and the modified Simplex Method for all function variables, the algorithm then determines if current model meets specifications. The algorithm checks to ensure that all the variables meet the requirements for both the p-value of each variable and the total number of variables. If the current model fails to meet the two requirements, the algorithm removes one variable. The algorithm uses two different methods to remove variables, one coupled with variables not meeting the p-value threshold and the other with having more than 12 total variables, all of which meet the p-value threshold.

To determine which variables the algorithm considers for removal, we use the Bonferroni Method to determine the p-value threshold for each individual variable. To determine the threshold, we divide 0.005 by the total number of variables currently in the algorithm (Neter et al., 1996). When one or more variables portray a failing p-value, greater than 0.005 divided by the total number of variables, the algorithm only considers removing those variables with failing p-values. The algorithm removes the variable with the least impact to the model. The algorithm uses the standardized Beta coefficient, previously described in the Literature Review Chapter, to determine the variables impact on the model. The algorithm sorts variables, only those with failing p-values, by the absolute standardized coefficient. The algorithm then selects the smallest standardized coefficient, of the variables with failing p-values, and removes the variable.

If all the variables show passing p-values, but the model contains more variables than 12 variables, then the algorithm must remove a variable. The algorithm will then sort all of the variables by their respective absolute standardized coefficient. After the sorting, the algorithm selects the smallest absolute standardized coefficient and removes the variable associated with it.

After the algorithm removes a variable, the algorithm re-runs the Cutting the Plane

algorithm and then the Simplex Method to re-optimize the thresholds for the variables, since removing variables could change the optimum thresholds. The algorithm will stop removing variables and optimizing when the model contains 12 or less variables, including the intercept, where each variable portrays a passing p-value, excluding the intercept. Once the algorithm meets the requirements, the algorithm determines the optimum order to run the Cutting the Plane algorithm and the Simplex Method. To determine the order, the algorithm runs through all permutations of the order of the variables the algorithm optimizes to determine the lowest possible SSE.

Determine Optimum Optimization Order

Once the algorithm selects the best combination of variables, it must determine the order to optimize the thresholds associated with those variables. The algorithm does not change the optimization order until after it selects the variables because of computational limitations. For example, in our analysis we use 24 (some variables contain more than one threshold) different thresholds for different variables. To determine the optimum order for those 24 variables, the algorithm runs the optimization procedures 6.2×10^{23} times. However, when six thresholds exists the algorithm. It only needs to run the procedures 720 times. To determine the optimum order, the algorithm runs the Cutting the Plane and Simplex Method multiple times where a new variable is optimized first each time. To determine the variable that the algorithm optimizes first, it compares the SSE at the end of each run. The algorithm then locks the variable displaying the lowest SSE in the first position. After determining the first position, the algorithm uses the same procedure to determine the remaining positions. Once the algorithm sets the order, it determines the variables' thresholds and determines the final parameter estimates for each variable.

Minimizing the MAPE

Finally, after the algorithm determines the thresholds and the significant variables, we generate the optimum coefficients by minimizing the MAPE. Using the MAPE as the loss function instead of the SSE ensures we do not over-weight a few data points compared to minimizing the SSE. Squaring the error term, when minimizing the SSE, is the cause for the over-weighting; however, when minimizing the MAPE the model does not square the error term, which leads to weighting all error equally. Minimizing the MAPE, after using the SSE for variable selection, could affect the variables significance. To ensure the accuracy of the data set used for analysis, we compare the accuracy of the model to the validation data set; we address this in further detail later in this chapter. After establishing our optimum coefficients, we use the model outputs to produce probabilities of problem occurrence.

Generating Control Chart Bounds

We run the algorithm to predict the EAC ratio for four, five, and six months into the future. We use these outputs to generate control charts. If a prediction from the model falls outside the bounds of the control chart, we consider this an indicator that a problem will occur within a given time period. We generate a control chart with two bounds, an upper bound and a lower bound. If a problem occurs within six months of detection, we identify that individual problem; however, if no detections occur within six months of a problem, we do not detect that problem.

To determine the bounds of the control chart, we optimize the percent of total problems the control chart detects while ensuring less than 30 percent of our predictions fall outside the control chart bounds. For example, if 50 potential problems exist in our four-month predictions, and we detect 30 of the 50 problems within six months of occurrence, while detecting less than

30 percent of the time, our optimization function displays a value of 0.6. We maximize this function by changing the upper and lower bounds of the control charts. The model that produces the highest optimization function and performs well with the validation data set we establish as our model for problem detection.

To optimize the percent of problems we detect, we change the control chart bounds. We use a complete grid search of every possible combination of control chart bounds to establish the optimum bounds. We compare the optimization function, as previously defined, of each different set of control chart bounds and select the bounds that produce the greatest value of the function. To perform the grid search, we use Crystal Ball® and set the upper and lower control chart bounds to uniform random variables. We set the upper control chart bound to a random number between zero and fifteen percent EAC growth from current month to the predicted month, we set the lower control chart bound to a random number between zero, and 10 percent EAC decrease. For example, on one trial, the bounds could display values of five percent growth and four percent decrease. With those bounds, any prediction where the EAC prediction increases greater than five percent or the EAC prediction decreases greater than four percent we deem it a detection. In that same scenario, any prediction of less than five percent EAC growth and four percent EAC decrease, we deem a non-detection. After each trial, we report the optimization function and after we run more than one million trials, we obtain the complete grid of possible combinations of control chart bounds. We do this procedure for our four through six-month model predictions and compare the aforementioned optimization function between the different control charts. We select the model prediction that detects the greatest percent of problems and use those predictions to determine probability of a problem occurrence.

After determining which model predictions we use, we make six different control charts,

with the bounds established from the optimization, to determine the probability a problem exists one through six months into the future. For example, if 10 predictions occur outside the control chart bounds and eight problems transpire within five months of those predictions, then the model states an 80 percent chance of a problem occurring within five months of a prediction falling outside the control chart bounds. We use these same procedures to establish probabilities of a problem occurrence one to six-months from a detection.

Validation

We use our validation set to test two things. First, we determine whether the point estimates provided by our three models prove statistically significant. Second, we test to ensure our bounds for the control chart demonstrate statistical significance. To ensure the point estimate's validity, we use a difference of means t-test, not assuming equal variances or population size, and determine the confidence level for the MAPE. We perform a one-tailed t-test where the null hypothesis states that the analysis data set's MAPE is greater than the MAPE of the validation's data set. We perform a one-tailed test because we only care if the MAPE increases for the validation set. If the p-value, for the difference of means test, demonstrates a value less than 0.1, we consider it a significant statistical difference between the means. Therefore, to pronounce no statistical difference between our validation set and the data we use to create our models, the p-values must be greater than 0.1. For the control chart, we perform a one-tailed difference of proportions z-test for the percent of time our control chart produces a correct detection or non-detection for the six-month estimate. We define a correct prediction as a non-detection when no problem occurs within six-months or a detection when a problem occurs within six-months. We test the null hypothesis that the proportion of correct predictions to total predictions for the analysis data set is less than the proportion of correct predictions to total

predictions for the validation data set. We use the same p-value thresholds for the difference of proportion test as we do for the difference of means test. After we ensure our data's validity, we compare our results to the current community's standard.

To determine the usefulness of our results to the acquisitions community, we compare our findings to a typical detection method. We use Keaton's (2011) detection algorithm and compare detection rates and accuracy rates. We feel Keaton's detection algorithm is representative of the typical tools an EVM expert uses in the field. If our results improve upon his and they pass the validation tests, we deem our findings both valid and useful. In the next chapter, Results Chapter, we assess our results from our EAC predictions, control charts, and the validation tests.

IV: Results

This chapter provides the results of the three forecasting models as well as our control chart for problem detection. We present the three formulas we use to make our EAC predictions for four, five and six months into the future. We address the accuracy and the shortfalls of our forecasts as well as our problem detection using the control chart.

The data is comprised of 67 months with absolute changes in the EAC from one month to the next greater than five percent. Nine of the changes, or problems, occur in our validation data. For the four-month control chart, seven of the 58 problems fall outside our eligible prediction window, nine and 11 for the five and six month control charts. We lose possible problems to detect because we do not use the first two months of data; additionally, we lose one month of data, within each program, for every extra month we predict. For example, our five-month prediction model contains 29 less months of data than our four-month prediction model since we use 29 programs in our analysis data set. Our four-month control chart detects 70 percent of the problems, the five-month control chart detects 73 percent of the problems, and the six-month control chart detects 74 percent of the problems. We address later in the chapter why we recommend using the four-month predictions for the control chart in lieu of not producing the optimal percent of problems detected.

Model Predictions

To reduce the complexity of our models, we limit the number of variables we use to predict the EAC. In our five and six-month models, the algorithm selected 11 variables, while the algorithm selected 10 variables for the four-month model. Reference Tables 4.1, 4.2, and 4.3 for a list of our equations and the results of the models we use to predict the EACs. Refer to

Appendix E for a description of the variables.

Our models predict the ratio of either the four, fifth, or sixth month divided by the current month EAC. To determine the point estimate of the fourth, fifth or sixth month EAC the analyst multiplies the ratio by the current month's EAC. For example, if the five-month model outputs a ratio of 1.0421 and the contractor reports an EAC of 143,000, then the model predicts a point estimate of 149,020.3. These estimates prove significant because of the results of statistical tests, shown in Tables 4.2, 4.3, 4.4.

Our three models we use to forecast the future EAC all pass validation. Table 4.1 displays the p-values associated with the difference of means test we perform, which we describe in the Methodology Chapter. All of the MAPE values for the validation set prove more accurate than the data we use to determine our variables. Refer to Table 4.1 for the results of the difference of means tests as well as the respective MAPE's for our models.

Table 4.1: Results of Prediction Models

	4-Month	5-Month	6-Month
P-Value	0.787	0.201	0.888
$MAPE_{analysis}$	3.135	3.675	4.080
$MAPE_{validation}$	2.695	3.551	3.442
$Sample\ Size_{analysis}$	861	832	803
$Sample\ Size_{validation}$	212	208	204

None of the model's error terms portray a normal distribution or constant variance; however, failing these two assumptions does not affect our models' predictions or their use in the control charts because we only use the point estimates generated from the models. Figure 4.1 displays histograms of each of our three models' studentized error distributions. We believe the few extreme, more than four standard deviations, prediction errors cause the deviation from normality. The drastic deviations from normality happen when the EAC changes by very large levels, greater than 30 percent. When the EAC changes by greater than 30 percent, our models

Table 4.2: Equation for Four-Month Prediction Results

Variable	Parameter Estimate (MAPE)	Percent Impact (OLS Standard Beta)	p-value (OLS)
Intercept	2.312		1.4E-22
CPI	-1.275	14.125%	1.2E-14
SPI	-1.159	10.084%	1.4E-13
SCI	1.123	17.368%	4.5E-14
Percent Difference Between ML and B	0.017	1.072%	2.5E-09
EAC Prediction CPI w/ no EAC Change	0.009	19.756%	2.0E-11
EAC Prediction Composite w/ no EAC Change	-0.010	18.745%	1.4E-10
EAC Prediction CPI w/ EAC Change	0.049	8.420%	8.5E-05
EAC Prediction Composite w/ EAC Change	-0.042	7.606%	1.8E-04
CPI Large w/ EAC Change	-0.043	1.170%	8.8E-07
Large Percent Difference Between B and W w/ EAC Change	0.141	1.654%	7.8E-17

Table 4.3: Equation for Five-Month Prediction Results

Variable	Parameter Estimate (MAPE)	Percent Impact (OLS Standard Beta)	p-value (OLS)
Intercept	2.7877		1.8E-46
CPI	-1.6890	22.030%	9.2E-40
SPI	-1.6161	15.931%	3.6E-33
SCI	1.5266	27.838%	2.0E-36
Percent Difference Between ML and B	0.1397	1.659%	1.3E-22
EAC Prediction CPI w/ no EAC Change	0.1140	14.294%	1.2E-07
EAC Prediction Composite w/ no EAC Change	-0.1241	14.104%	1.7E-07
CPI Large w/ EAC Change	-0.0106	1.001%	7.7E-09
CPI Small w/ EAC Change	0.0233	0.606%	1.3E-04
TSPI Large w/ EAC Change	-0.0382	0.873%	1.8E-10
Large Percent Difference Between B and W w/ EAC Change	0.1136	0.942%	3.8E-08
Small percent Difference Between ML and B w/ EAC Change	0.0053	0.723%	2.9E-09

Table 4.4: Equation for Six-Month Prediction Results

Variable	Parameter Estimate (MAPE)	Percent Impact (OLS Standard Beta)	p-value (OLS)
Intercept	2.150		1.3E-20
CPI	-1.203	12.809%	2.4E-16
SPI	-0.996	9.580%	7.3E-15
SCI	1.043	17.176%	2.1E-17
Percent Difference Between ML and B	-0.006	1.320%	4.0E-16
EAC Prediction CPI w/ no EAC Change	0.038	14.183%	1.6E-07
EAC Prediction Composite w/ no EAC Change	-0.025	12.449%	4.7E-06
EAC Prediction CPI w/ EAC Change	0.154	15.163%	3.9E-12
EAC Prediction Composite w/ EAC Change	-0.133	13.824%	1.7E-11
CPI Small w/ EAC Change	0.017	0.713%	5.7E-06
SCI Large w/ EAC Change	-0.067	1.674%	2.0E-16
Large percent Difference Between B and W w/ EAC Change	0.150	1.109%	8.4E-11

do not predict the magnitude of the change accurately. However, our models typically predict a change great enough to indicate a problem; we address problem indication later in the chapter. Therefore, even though the model diverges in accuracy, it still provides the correct information to decision makers.

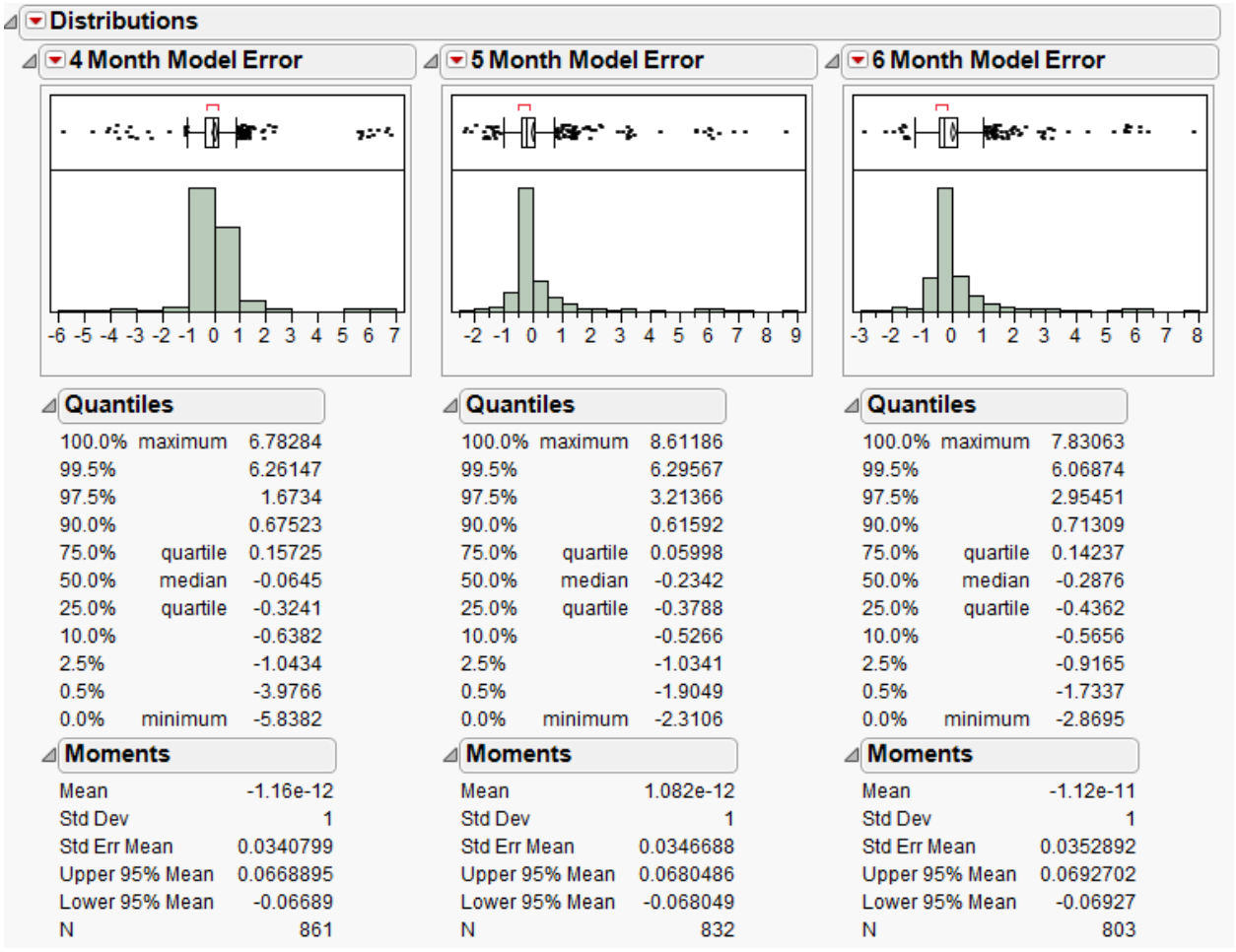


Figure 4.1: Histograms of Model Error

Since we only use the point estimates of our model and do not use a confidence interval, it proves unnecessary for the error term to contain constant variance. We made our overall p-value for excluding variables 0.005 to ensure the significance of the variables we select before minimizing for the MAPE. All of the Cooks Distances for each of the models present values lower than 0.5; therefore, we conclude that none of our monthly observations overly influences

the coefficients in our models. Reference Appendix G for our Cooks Distance charts.

Due to the strong performance with the validation set, we feel confident that these few problems with the model assumptions do not affect our models use. All three of the models' predictions present lower MAPE scores on the validation data than the data we use to generate our Parameter estimates; Tables 4.1, 4.2, and 4.3 display the results of our models. The three models also passed validation, the p-value for the t-test displayed a value greater than 0.1. These p-values reinforce our confidence in our models' predictions and the use of those predictions in our control charts. We present a breakout of the absolute percent error (APE) of each model in Figure 4.2.

Figure 4.2: Histograms of Model APE's

Control Chart

Our control charts provide decision makers with the likelihood of a problem occurring within a given time period. We use the z-test to test the percentage of time the control chart produces a correct prediction to validate our control chart bounds. Table 4.5 presents the results of each of the control charts and their performance in the difference of proportions z-test. Our validation data contains a limited number of problems to detect, but this does not affect our validation of our control charts. Since we use the percent of time the control chart provides correct predictions to validate our data, the percent of total problems the control charts detect does not change our validation of the models. The limited number of possible problems to detect in our validation data limits our analysis on this statistic for the validation data; therefore, we do not compare the percentage of problems the control chart detects in the validation set to the data set we use for our analysis.

We use our four-month control chart to provide the likelihood of a problem occurrence. Our four-month control chart does not detect as many, six percent less, of the overall problems as the six-month control chart; however, it does detect more of the problems in the validation data. The five-month control chart detects two of the problems and the six-month control chart only detects one problem. The four-month control chart has the opportunity to detect five problems while the five and six-month control charts only have the opportunity to detect four problems. The loss in data due to the forecasting period causes the decrease in the detection opportunity. The four-month control chart presents more correct predictions than both the five and six-month control charts in both the analysis data and the validation data. Since the four-month control chart performs better with the validation data set and only a small difference exists in the data we use in our analysis, we use the four-month control chart to determine likelihoods.

Table 4.5: Control Chart Results

	4-Month Prediction	5-Month Prediction	6-Month Prediction
Upper Control Chart Bound	1.0146	1.0212	1.0211
Lower Control Chart Bound	0.9787	0.9810	0.9772
% of Time Detection Occurs (analysis)	28.80%	29.44%	29.27%
% of Time Detection Occurs (validation)	21.00%	32.55%	15.57%
% of Time Correct (analysis)	71.15%	69.35%	69.61%
% of Time Correct (validation)	74.50%	67.90%	73.58%
% of Total Problems Detected (analysis)	70.00%	73.50%	75%
% of Total Problems Detected (validation)	40.00%	50.00%	25.00%
p-value for Z-test (proportion of analysis<proportion of validation	0.834	0.343	0.866
Sample Size$_{analysis}$	861	832	803
Sample Size$_{validation}$	212	208	204

Using the four-month control chart, we determine the percentage of total problems the control charts detect within different time periods. In addition, we determine the probability of a problem occurrence given a detection and the probability that a problem will not occur given that we do not detect a problem. Table 4.6 displays these likelihoods. We graph all of our correct and incorrect predictions, of a four-month control chart, using a scatter plot with our control chart bounds to provide a visual representation of our data. See Figure 4.3 for the control chart. Figure 4.4 depicts a zoomed in control chart portraying data points 450-549. In both control charts, grey depicts an incorrect prediction and black depicts a correct prediction. A black dot

falling outside the bounds means we detect a problem and a problem occurs within six months, while a black dot within the bounds means we do not detect a problem and no problem occurs within six months. A grey dot outside the bounds means we detect a problem and no problem occurs within six months, while a grey dot within the bounds means we do not detect a problem and a problem does occur within six months.

Table 4.6: Breakout of Probabilities

	Within 1 Month of Occurrence	Within 2 Months of Occurrence	Within 3 Months of Occurrence	Within 4 Months of Occurrence	Within 5 Months of Occurrence	Within 6 Months of Occurrence
Percent of Total Problems Detected	48.00%	52.00%	58.00%	64.00%	64.00%	70.00%
Probability of a Problem Given a Detection	11.06%	19.82%	29.03%	34.56%	40.09%	42.34%
Probability of No Problem Given No Detection	96.59%	93.01%	90.08%	86.83%	84.55%	83.73%

In our data set, when the four-month model predicts extremely high, greater than 1.14, or extremely low, less than 0.935, a problem always occurs within six-months of that point. In our model, we do not see a relationship between successive detections and the likelihood of problem occurrence. Table 4.7 displays the results of our control chart compared to one method the DoD acquisition's community currently uses, Keaton's (2011) one standard deviation CPI detection algorithm. The boxes in the table portray the conditional probabilities given a detection or non-detection. For example, the top left box exhibits the probability of a problem in six months given a detection, while the lower left box in the table depicts the probability of no problem occurring within six months of a detection. The right column displays the same values except given a non-detection instead of a detection as the state of nature. The top right box represents the false negatives and the bottom left represents the false positives. Our method improves on

Keaton's (2011) method in both false positives and false negatives. In the next chapter, we discuss the implications of our findings as well as future areas to improve upon our research.

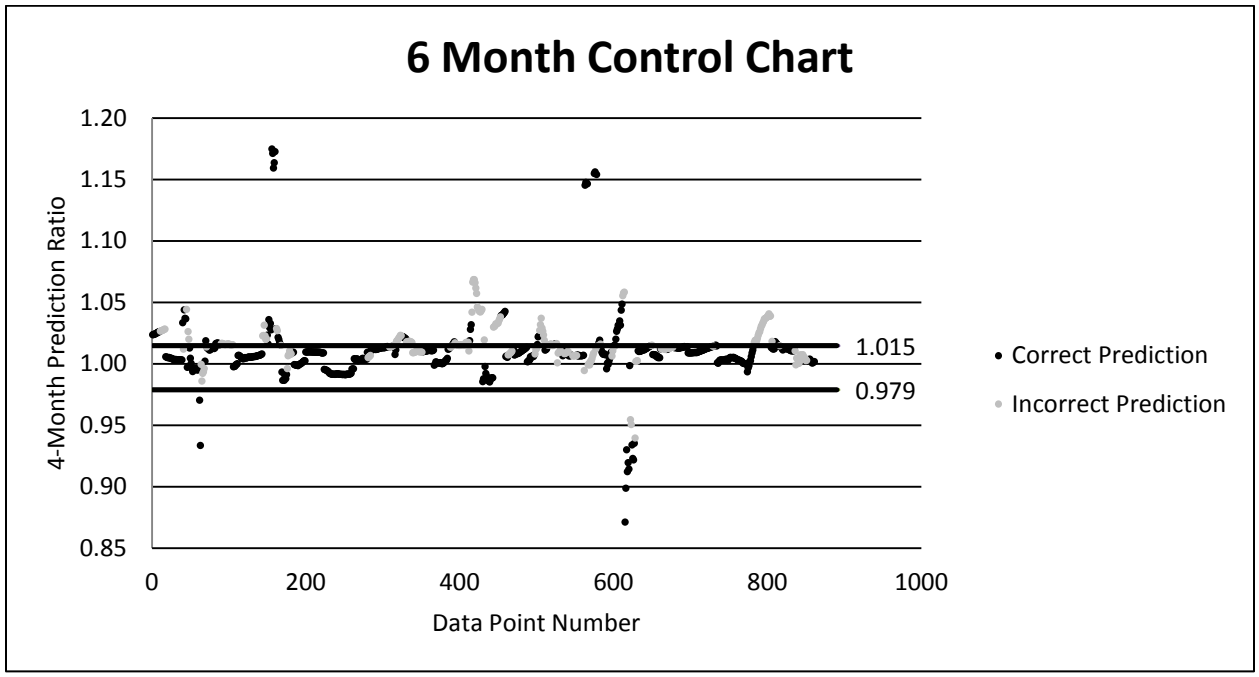


Figure 4.3: 6-Month Control Chart Using Four-Month Predictions

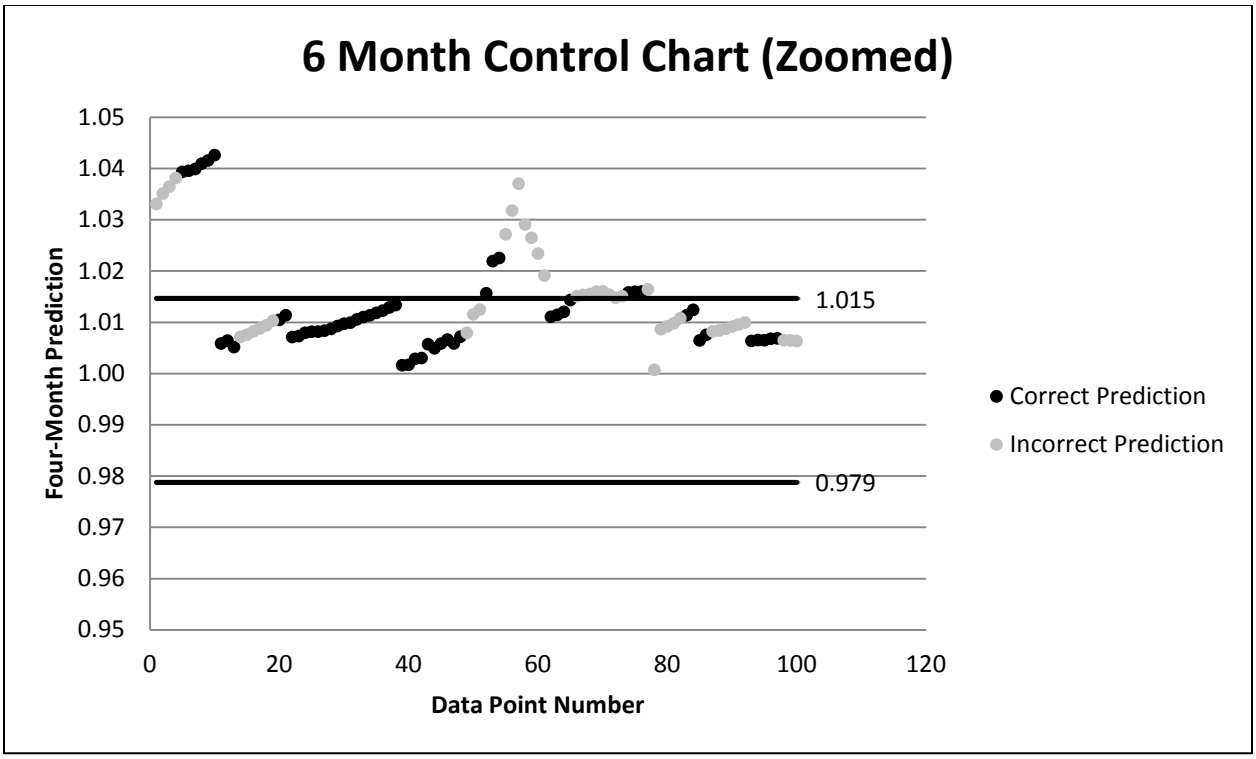


Figure 4.4: Control Chart Using Four-Month Predictions Zoomed

Table 4.7: Comparison of Our Results to Community Standard

Our Method		
	Detection	**No Detection**
Problem	42.34%	16.27%
No Problem	57.66%	83.73%

Community Standard (Keaton's 1 Stdev CPI Method)		
	Detection	**No Detection**
Problem	22.69%	28.00%
No Problem	77.31%	72.00%

V: Conclusions

Discussion of Results

We set out to answer a few initial research questions:

1. Can we provide an accurate point estimate for future contractor provided EAC's?
2. Can we detect future major changes to the EAC?
3. If we detect major changes to the EAC, can we provide decision makers with a timeframe and probability of those major changes to the EAC?

We answer the first question by providing three models that predict the contractor provided most likely EAC four, five, and six months into the future. We develop these predictions through an optimization algorithm. We find our optimization algorithm provides three sufficient models to provide decision makers with a point estimate of the EAC six months from the current period within an average of four percent. These predictions feed into our control charts to answer the last two research questions.

Our control charts detect 70 percent of the total problems while only identifying 28 percent of the months as potential problems. We detect more overall problems than the previous researcher's models (Keaton et al., 2011), while producing less false positive detections. Our control charts provide accurate predictions of either a future problem, or no future problem, over seventy percent of the time. These results provide decision makers with essential information as to when a problem might occur as well as its probability.

As with previous research, determining what represents a program problem actually presents itself as a problem. To overcome this issue, we use the same definition of a problem as the previous research. This ensures continuity between our research and allows us to baseline our results against the previous examination.

Our algorithm outperforms prior researcher's model (Keaton et al., 2011) by nearly a factor of two in reference to the accuracy of a control chart detection. For example, if a decision maker controls 20 programs and uses our method to determine if a problem will occur in their program, our algorithm will detect five programs while the Keaton model will detect eight programs. Two problems will exist within our algorithm's five detections. In contrast, the same two problems will exist within Keaton's model, but his model requires excessive detections (in this example, eight). This added accuracy allows our algorithm to enhance the oversight to acquisition programs. The higher level of accuracy enables DoD leadership to better allocate their resources and prevent future acquisition problems. The early detection should prevent programs from remaining unstable past the 20 percent completion. We believe if program managers implement our detection algorithm procedures at the start of their program, the likelihood of their program going over budget will decrease.

Our research does contain a few areas of concern; however, we feel these concerns do not limit the validity and reliability of our findings. Our validation data set limits our problem detection since only five potential problems exists for our control chart to detect. However, we overcome this issue by comparing the overall accuracy of the control charts. To compare the overall accuracy, we compare the percent of correct predictions between our different data set. In our validation data set, we detect just over 20 percent of the time, while with the data we use for analysis we detect close to 28 percent of the time. These differences ensure the accuracy and validity of our control charts because more potential problems exist within the data set we use for analysis; therefore, it should detect more frequently. Additionally, the closeness between the accuracy of our analysis data and our validation data predictions reaffirms our confidence in our results.

Our high level of accuracy for our point estimates proves a testament as to the quality of our data-mining algorithm that we previously described in the Methodology Chapter. Our algorithm institutes a highly effective procedure for determining relationships and generating variables within a data set. Since our algorithm does not need to use a specific type of data, researchers and analysts can use it to generate models for any type of data. The procedures we establish in the Methodology Chapter serve as a way for analysts to provide leadership with the information they require to make informed decisions.

Implications of Findings

Providing decision makers with the probability and timing of a future problem occurrence, enables them to focus on the DoD contracts that show early signs of poor performance. This early detection will hopefully prevent future problems and save the DoD millions of dollars in cost overruns. These potential problems also affect the contract schedule, and the early detection enables DoD leaders the opportunity to provide more oversight and reduce the amount of future schedule slips. The point estimates we provide allows DoD leadership to compare between contracts to determine which one(s) needs the most attention.

These estimates enable leadership to track and forecast the course of the program. The point estimates also serve as a way to distinguish between multiple detections. For example, if a decision maker controls 20 programs and our algorithm detects five programs where a potential problem will exists, the point estimate serves as a comparison of which program needs the most attention. If three of the five predictions predict a three percent increase in the EAC in four months and two predict an increase of five percent, a decision maker can address the two with the greater prediction first. In addition, the point estimates serve as a way of checking the algorithm's accuracy within their program. Since we combine multiple acquisition programs

together within our analysis, some programs demonstrate more accurate results than others. A decision maker can take that information to determine if they want to use our algorithm within their programs, which will increase their management efficiency.

Early problem detection increases efficiencies in DoD programs. The detection of problems provides the capability to better utilize personnel. With an algorithm to determine when potential problems occur, acquisition personnel will not spend their time consumed with tracking program data. The acquisition personnel will spend their extra time performing their primary duties. In a time of DoD downsizing, a process that automates redundant work increases the overall capability of the acquisitions community.

Follow on Research

Using our method to determine when a possible problem will occur, permits decision makers to focus on the programs that require the most attention. Our research does not provide a decision maker the area within the contract that causes the potential problem. Future research can use lower level CPR's to determine the cause of the potential problem. Determining this cause will enable program managers to spotlight the area that needs the most attention.

In addition, follow on researchers can apply our methods to non DoD contracts. Contract management for commercial construction companies or the Department of Energy requires close project management as well. In this study, the research could focus on changes to the sensitivity of the detection and possibly a control chart with non-stationary bounds. A control chart with non-stationary bounds would decrease the false positives of our research while maintaining the overall effectiveness of our detections. A non-stationary control chart could use text-mining input to determine the level of deviation in the EVM data required to indicate a potential problem.

Appendix A: Example Format-1 (AEHF Program)

COST PERFORMANCE REPORT
FORMAT 1 - WORK BREAKDOWN STRUCTURE — DOLLARS IN: Thousands — Page 1 of 3

1. CONTRACTOR	2. CONTRACT	3. PROGRAM	4. REPORT PERIOD
a. NAME LOCKHEED MARTIN SPACE SYSTEMS	a. NAME AEHF	a. NAME AEHF SDD	a. FROM (CCYYMMDD) 20071001
b. LOCATION (Address and ZIP code) 1111 LOCKHEED MARTIN WAY	b. NUMBER F04701-02-C-0002		b. TO (CCYYMMDD) 20071028
SUNNYVALE, CA USA 94088	c. TYPE CPAF — d. SHARE RATIO 100/0 0/100	b. PHASE (X one) x RDT&E ☐ PRODUCTION	

5. CONTRACT DATA							
a. QUANTITY	b. NEGOTIATED COST	c. EST COST AUTH UNPRICED WORK	d. TARGET PROFIT/ FEE	e. TARGET PRICE	f. ESTIMATED PRICE	g. CONTRACT CEILING	h. ESTIMATED CONTRACT CEILING
PROD: 0 R&D: 3	$3,883,652.6	$17,714.5	$458,544.2 / 0.0%	$4,342,196.7	$4,838,032.0		

6. ESTIMATED COST AT COMPLETION	MANAGEMENT ESTIMATE AT COMPLETION (1)	CONTRACT BUDGET BASE (2)	VARIANCE (3)
a. BEST CASE	$4,364,262.0		
b. WORST CASE	$4,480,862.0		
c. MOST LIKELY	$4,377,362.1	$3,901,367.1	$-475,995.0

7. AUTHORIZED CONTRACTOR REPRESENTATIVE		
a. NAME (Last, First, Middle Initial) TRAN, J.	b. TITLE CONTRACTS	
c. SIGNATURE		d. DATE (CCYYMMDD) 20071203

8. PERFORMANCE DATA

ITEM		CURRENT PERIOD: BUDGETED COST WORK SCHEDULED (2)	BUDGETED COST WORK PERFORMED (3)	ACTUAL COST WORK PERFORMED (4)	VARIANCE SCHEDULE (5)	VARIANCE COST (6)	CUMULATIVE TO DATE: BUDGETED COST WORK SCHEDULED (7)	BUDGETED COST WORK PERFORMED (8)	ACTUAL COST WORK PERFORMED (9)	VARIANCE SCHEDULE (10)	VARIANCE COST (11)	REPROGRAM ADJUSTMENTS: COST VARIANCE (12)	REPROGRAM ADJUSTMENTS: BUDGET (13)	AT COMPLETION: BUDGETED (14)	AT COMPLETION: ESTIMATED (15)	AT COMPLETION: VARIANCE (16)
a. WBS ELEMENT (1)																
TOTAL COST - AEHF SYSTEM DESIGN AND DEVELOPMENT	2	38,161	40,147	51,861	1,987	-11,714	3,175,890	3,151,116	3,506,736	-24,774	-355,620			3,882,108	4,345,003	-462,895
SV 1&2 - SPACE VEHICLE 1&2	3	28,184	29,301	40,158	1,116	-10,857	2,853,424	2,838,185	3,202,538	-15,240	-364,354			3,337,430	3,803,096	-465,667
1.0 - SPACE VEHICLE	4	12,551	13,148	23,298	597	-10,150	1,948,792	1,939,698	2,323,748	-9,094	-384,050			2,106,373	2,584,411	-478,038
1.1 - SPACECRAFT BUS	5	2,212	2,221	4,826	9	-2,605	271,192	266,255	344,754	-4,937	-78,499			292,007	384,538	-92,532
1.1.1 - STRUCTURES/PROP/THERMAL HDWE	6	134	295	869	161	-574	46,711	45,932	70,066	-779	-24,133			48,453	75,377	-26,924
1.1.2 - GUIDANCE NAVIGATION & CONTROL	6	56	56	49	0	7	16,282	16,145	18,835	-137	-2,689			16,392	19,568	-3,176
1.1.3 - SOLAR ARRAYS & MECHANISMS	6	183	313	1,364	130	-1,051	37,956	36,295	50,236	-1,661	-13,941			39,038	56,917	-17,878
1.1.4 - HIGH POWER ELECTRONICS	6	939	522	861	-416	-339	32,116	30,971	35,202	-1,144	-4,231			33,522	39,028	-5,505
1.1.5 - TELEMETRY TRACK & CONTROL HDWE	6	0	0	2	0	-2	12,994	12,994	14,087	0	-1,093			12,994	14,097	-1,103
1.1.6 - COMMAND/DATA HANDLING HDWE	6	0	76	232	76	-156	40,800	40,597	51,376	-203	-10,779			40,800	51,856	-11,056
1.1.7 - SPACECRAFT BUS FLIGHT SOFTWARE	6	553	527	619	-26	-92	38,859	38,150	45,138	-709	-6,988			46,104	55,492	-9,388
1.1.8 - SPACECRAFT BUS SE/PM	6	316	317	709	1	-392	29,301	29,298	42,132	-3	-12,834			37,968	53,541	-15,573
1.1.9 - SPACECRAFT BUS I&T	6	30	114	120	84	-6	16,173	15,872	17,683	-301	-1,811			16,735	18,662	-1,927
1.2 - EHF PAYLOAD	5	8,846	9,613	13,881	767	-4,268	1,606,935	1,605,127	1,881,821	-1,808	-276,694			1,711,172	2,043,550	-332,378
1.2.7 - PAYLOAD 1-17-19-21-23-25-27-45	6	8,846	9,613	13,881	767	-4,268	1,606,935	1,605,127	1,881,821	-1,808	-276,694			1,711,172	2,043,550	-332,378
1.3 - LAUNCH SUPPORT OPERATIONS	5	245	77	27	-169	50	2,160	1,914	1,190	-246	724			11,181	10,750	431
1.4 - SPACE VEH AGE/MAGE	5	28	61	1,147	34	-1,086	8,690	8,573	15,661	-117	-7,088			9,152	20,048	-10,896

COST PERFORMANCE REPORT
FORMAT 1 - WORK BREAKDOWN STRUCTURE

DOLLARS IN: Thousands

8. PERFORMANCE DATA

ITEM		CURRENT PERIOD					CUMULATIVE TO DATE					REPROGRAM ADJUSTMENTS		AT COMPLETION		
		BUDGETED COST		ACTUAL	VARIANCE		BUDGETED COST		ACTUAL	VARIANCE						
		WORK SCHEDULED	WORK PERFORMED	COST WORK PERFORMED	SCHEDULE	COST	WORK SCHEDULED	WORK PERFORMED	COST WORK PERFORMED	SCHEDULE	COST	COST VARIANCE	BUDGET	BUDGETED	ESTIMATED	VARIANCE
(1)		(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)
a. WBS ELEMENT																
1.5 - SPACE VEH SE/PM	5	422	410	919	-12	-509	45,604	45,576	53,491	-28	-7,915			52,877	69,022	-16,145
1.6 - SPACE VEH I&T	5	798	766	2,498	-31	-1,731	14,212	12,253	26,832	-1,959	-14,579			29,984	56,503	-26,518
2.0 - MISSION CONTROL SYSTEM	4	8,263	9,035	9,100	772	-65	550,983	549,606	549,347	-1,377	259			682,371	679,262	3,109
2.1 - MOPS 2-6-7-8-9-11-13-14-28-33-34-44	5	8,094	8,867	8,857	773	10	534,447	533,099	533,891	-1,348	-792			664,058	661,827	2,231
2.6 - MCS INTEGRATION & TEST - 10	5	0	0	0	0	0	294	294	301	0	-6			294	301	-6
2.7 - MCS SE/PM 3-4-5-12-15	5	169	168	243	0	-75	16,241	16,213	15,156	-29	1,057			18,019	17,134	884
3.0 - INTERSEGMENT SYS ENG/PGM MGMT	4	4,105	4,143	4,215	38	-73	239,578	237,168	227,633	-2,410	9,535			336,640	333,344	3,296
3.1 - SYSTEM ENGINEERING - 16	5	1,174	1,144	1,152	-30	-9	69,687	69,105	62,258	-582	6,847			105,697	97,427	8,270
3.2 - PROGRAM MANAGEMENT - 18	5	2,378	2,446	2,690	68	-245	150,695	149,075	148,708	-1,621	367			197,777	207,689	-9,912
3.3 - SYSTEM DATA BASE	5	553	553	373	0	180	19,195	18,988	16,667	-207	2,321			33,166	28,228	4,938
6.0 - INTERSEGMENT/SYSTEM LEVEL I&T	4	2,273	1,977	2,645	-296	-667	89,588	87,225	83,087	-2,363	4,138			175,837	175,540	297
6.1 - SYSTEM TEST EQUIPMENT - 20	5	493	493	931	0	-438	36,577	35,440	36,555	-1,137	-1,115			58,368	61,831	-3,463
6.2 - FACTORY SYSTEM LEVEL TEST - 22	5	692	550	776	-142	-227	20,308	19,880	18,140	-428	1,739			35,682	34,863	819
6.3 - EARLY ORBIT OPERATIONS - 24	5	959	806	838	-154	-32	29,887	29,144	26,181	-743	2,963			64,648	61,700	2,948
6.4 - ON-ORBIT TEST - 26	5	129	129	100	0	29	2,817	2,761	2,211	-55	551			17,139	17,146	-7
7.0 - OPERATIONS & SUPPORT	4	831	833	716	2	116	6,948	6,949	5,739	2	1,210			17,241	16,148	1,092
7.2 - SUSTAINING SUPPORT 29-30	5	36	36	2	0	34	265	265	105	0	161			848	722	126
7.3 - INTERIM MAINTENANCE 31-32	5	795	797	715	2	82	6,682	6,684	5,634	2	1,049			16,393	15,427	966
8.0 - SPECIAL STUDIES	4	162	165	184	3	-19	17,536	17,538	12,983	2	4,555			18,968	14,391	4,577
8.1 - INVESTIGATION & ANALYSIS	5	162	165	184	3	-19	17,536	17,538	12,983	2	4,555			18,968	14,391	4,577
SV 3 - SPACE VEHICLE 3	3	9,976	10,846	11,703	870	-857	282,966	273,432	264,698	-9,534	8,734			505,178	502,407	2,772
1.0A - SPACE VEHICLE	4	9,769	10,605	11,458	836	-853	277,492	268,496	260,496	-8,996	8,000			459,419	457,479	1,940
1.1A - SPACECRAFT BUS	5	1,196	2,244	2,138	1,049	107	59,329	52,072	49,951	-7,258	2,120			95,425	94,434	991
1.1.1A - STRUCTURES/PROP/THERMAL HDWE	6	57	1,014	1,195	956	-181	19,497	17,262	17,955	-2,235	-693			22,737	24,104	-1,367
1.1.2A - GUIDANCE NAVIGATION & CONTROL	6	314	104	103	-210	1	7,049	6,788	6,652	-261	136			7,801	7,792	10
1.1.3A - SOLAR ARRAYS & MECHANISMS	6	54	54	152	0	-98	9,993	7,484	7,399	-2,509	85			15,624	15,624	0
1.1.4A - HIGH POWER ELECTRONICS	6	343	348	-23	5	371	4,870	5,685	4,369	815	1,316			12,312	10,804	1,509
1.1.5A - TELEMETRY TRACK & CONTROL HDWE	6	0	0	16	0	-16	3,254	2,235	2,088	-1,019	147			3,254	3,135	119
1.1.6A - COMMAND/DATA HANDLING HDWE	6	119	63	225	-57	-163	11,281	9,262	9,276	-1,999	6			11,671	12,459	-787
1.1.7A - SPACECRAFT BUS FLIGHT SOFTWARE	6	70	70	54	0	16	220	220	177	0	43			5,629	5,911	-281
1.1.8A - SPACECRAFT BUS SE/PM	6	192	192	133	0	59	2,247	2,247	882	0	1,365			14,015	12,265	1,750
1.1.9A - SPACECRAFT BUS I&T	6	47	401	283	354	119	918	868	1,152	-49	-284			2,379	2,341	38
1.2A - EHF PAYLOAD	5	8,399	8,186	9,126	-213	-940	213,524	211,811	206,545	-1,713	5,266			332,260	331,376	884
1.2.7A - PAYLOAD 35-37-39-41-43	6	8,399	8,186	9,126	-213	-940	213,524	211,811	206,545	-1,713	5,266			332,260	331,376	884

COST PERFORMANCE REPORT
FORMAT 1 - WORK BREAKDOWN STRUCTURE

DOLLARS IN: Thousands

8. PERFORMANCE DATA

ITEM		CURRENT PERIOD					CUMULATIVE TO DATE					REPROGRAM ADJUSTMENTS		AT COMPLETION		
		BUDGETED COST		ACTUAL	VARIANCE		BUDGETED COST		ACTUAL	VARIANCE						
		WORK SCHEDULED	WORK PERFORMED	COST WORK PERFORMED	SCHEDULE	COST	WORK SCHEDULED	WORK PERFORMED	COST WORK PERFORMED	SCHEDULE	COST	COST VARIANCE	BUDGET	BUDGETED	ESTIMATED	VARIANCE
(1)		(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)
a. WBS ELEMENT																
1.4A - SPACE VEH AGE/MAGE	5	14	14	0	0	14	1,948	1,948	1,555	0	393			2,478	2,478	0
1.5A - SPACE VEH SE/PM	5	125	125	94	0	31	1,800	1,775	1,401	-25	374			12,081	12,016	65
1.6A - SPACE VEH I&T	5	35	35	101	0	-65	890	890	1,043	0	-153			17,176	17,176	0
3.0A - INTERSEGMENT SYS ENG/PGM MGMT	4	207	242	245	35	-3	5,474	4,936	4,202	-538	734			37,099	36,351	748
3.1A - SYSTEM ENGINEERING - 36	5	11	11	0	0	11	39	39	13	0	26			2,962	2,615	347
3.2A - PROGRAM MANAGEMENT - 38	5	196	230	245	35	-14	5,435	4,897	4,189	-538	708			29,502	29,218	284
3.3A - SYSTEM DATA BASE	5	0	0	0	0	0	0	0	0	0	0			4,635	4,518	117
6.0A - INTERSEGMENT/SYSTEM LEVEL I&T	4	0	0	0	0	0	0	0	0	0	0			8,661	8,577	83
6.1A - SYSTEM TEST EQUIPMENT	5	0	0	0	0	0	0	0	0	0	0			5,881	5,879	2
6.2A - FACTORY SYSTEM LEVEL TEST - 40	5	0	0	0	0	0	0	0	0	0	0			1,941	1,859	81
6.3A - EARLY ORBIT OPERATIONS - 42	5	0	0	0	0	0	0	0	0	0	0			839	839	0
UCA3 - UCA3	3	0	0	0	0	0	39,500	39,500	39,500	0	0			39,500	39,500	0
b. COST OF MONEY	N 2	159	168	283	9	-115	8,503	8,215	9,383	-288	-1,168			13,269	15,516	-2,248
c. GENERAL & ADMINISTRATIVE	N 2	1,759	1,851	2,797	92	-947	104,197	101,879	115,341	-2,318	-13,462			155,038	178,190	-23,152
d. UNDISTRIBUTED BUDGET	2													10,991	10,991	0
e. SUBTOTAL *(Performance Measurement Baseline)*		38,161	40,147	51,861	1,987	-11,714	3,175,890	3,151,116	3,506,736	-24,774	-355,620	0	0	3,893,099	4,355,994	-462,895
f. MANAGEMENT RESERVE	2												0	8,268		
g. TOTAL		38,161	40,147	51,861	1,987	-11,714	3,175,890	3,151,116	3,506,736	-24,774	-355,620	0	0	3,901,367		
9. RECONCILIATION TO CONTRACT BUDGET BASE																
a. VARIANCE ADJUSTMENT										0	0					
b. TOTAL CONTRACT VARIANCE										-24,774	-355,620			3,901,367	4,377,362	-475,995

Appendix B: EVM Equations (Keaton 2011)

Descriptive EVM Measures	Equation	Interpretation
Cost Variance (CV$)	$CV\$ = BCWP - ACWP$	Difference between value and cost of work accomplished
Normalized Cost Variance (NCV)	$NCV = \frac{CV\$}{BAC}$	Cost Variance relative to contract size
Percent Cost Variance (CV%)	$CV\% = \frac{CV\$}{BCWP} * 100$	Shows over and under budget
Schedule Variance (SV$)	$SV\$ = BCWP - BCWS$	Difference between value of work accomplished and value scheduled
Schedule Variance (SVMonths)	$SVMonths = \frac{SV\$}{BCWS}$	Provides a time value for work finished ahead and behind schedule
Normalized Schedule Variance (NSV)	$NSV = \frac{SV\$}{BAC}$	Schedule Variance relative to contract size
Percent Schedule Variance (SV%)	$SV\% = \frac{SV\$}{BCWS} * 100$	Shows ahead and behind schedule
Variance At Completion (VAC)	$VAC = BAC - EAC$	Difference between cost budgeted and cost estimated
Cost Performance Index (CPI)	$CPI = \frac{BCWP}{ACWP}$	Compares the budget to the amount of money spent
Schedule Performance Index (SPI)	$SPI = \frac{BCWP}{BCWS}$	Compares actual value to the value plan
Schedule Cost Index (SCI)	$SCI = CPI * SPI$	
Composite Index (CMI)	$CMI = \alpha CPI + \beta SPI$	
To Complete Performance Index ($TCPI_{EAC}$)	$TCPI = \frac{(BAC - BCWP_{CUM})}{(EAC - ACWP_{CUM})}$	Measures cost efficiency requirement to complete on-budget
Percent Complete (BAC)	$\%\text{Complete} = \left(\frac{BCWP_{CUM}}{BAC}\right) * 100$	Compares work plan to program budget
Percent Complete (Months)	$\%\text{Complete} = \left(\frac{Months\ from\ Start\ Date}{Total\ Months\ of\ Contract}\right) * 100$	Compares the amount of time spent for a contract to the total amount of time

Appendix C: Breakout of Data

Program	Months of Data	Validation (Yes/No)
B2-EHF	14	No
AMF JTRS SDD (BBX)	20	Yes
MM III GRP FRP '07	20	No
Non Line of Sight - Launch System (FCS Navy)	20	No
C130J BUIC Del Order 0003	22	No
LCS - CLIN 0008 AUSTAL	24	No
E-2D Advanced Hawkeye (AHE)	27	No
EFV SDD-2	27	Yes
B-2 RMP	28	No
FORCE XXI BATTLE COMMAND BRIGADE AND BELOW (FBCB2)	28	No
NPOESS	28	No
NMT EDM	30	No
C-130 Block 6.5.1 HCMC	31	No
E871209B (MH-60)	31	Yes
CH-53	32	Yes
V-22	33	Yes
WINT_INC2-M	33	No
ISPAN	34	No
MPS - FPM	37	No
UH-60M	37	No
WGS BLOCK II	37	No
MP-RTIP Phase 2	41	No
Blue Grass Chemical Agent Destruction Pilot Plant	42	No
DDG 1000	42	Yes
F-35 JSF System Development & Demonstration	42	No
Chem. Demil Stockp (Chem Demil CMA)	43	No
GPS MUE CLIN 002 (Navstar)	43	Yes
C130 Avionics Moderinzation Program	44	No
SBIRS	44	No
AEHF	45	No
C-5 Reliability Enhancement & Reengining Program SDD	45	No
MPEC JMPS-E (mps-exp ops)	45	No
SM6	45	No
MPS SEICR1	48	Yes
MOBILE USER OBJECTIVE SYSTEM (MUOS)	50	No
JLENS	52	No
P-8	52	No
		No

Appendix D: Complete List of Initial Variables

Variable Name
EAC Lag 1
EAC Lag 2
EAC Lag 3
CPI*Previous EAC
SPI*s EAC
TSPI* EAC
TCPI* EAC
SCI* EAC
SV%* EAC
CV%*EAC
(% Difference Between ML and W)* EAC
(% Difference Between ML and B)*s EAC
(% Difference Between W and B)* EAC
(StDev CPI)* EAC
(StDev SPI)* EAC
(TSPI StDev)* EAC
(TCPI StDev)* EAC
(SCI StDev)* EAC
(SV% StDev)* EAC
(CV% StDev)*EAC
EAC Prediction CPI w/ no EAC Change
EAC Prediction Composite w/ no EAC Change
EAC Prediction CPI w/ EAC Change
EAC Prediction Composite w/ EAC Change
CPI Large w/ no EAC Change
CPI Medium w/ no EAC Change
CPI Small w/ no EAC Change
SPI Large w/ no EAC Change
SPI Medium w/ no EAC Change
SPI Small w/ no EAC Change
SCI Large w/ no EAC Change
SCI Medium w/ no EAC Change
SCI Small w/ no EAC Change
TCPI Large w/ no EAC Change
TCPI Medium w/ no EAC Change
TCPI Small w/ no EAC Change
TSPI Large w/ no EAC Change

TSPI Medium w/ no EAC Change
TSPI Small w/ no EAC Change
SV% Large w/ no EAC Change
SV% Medium w/ no EAC Change
SV% Small w/ no EAC Change
CV% Large w/ no EAC Change
CV% Medium w/ no EAC Change
CV% Small w/ no EAC Change
StDev CPI Large w/ no EAC Change
StDev CPI Small w/ no EAC Change
StDev SPI Large w/ no EAC Change
StDev SPI Small w/ no EAC Change
StDev SCI Large w/ no EAC Change
StDev SCI Small w/ no EAC Change
StDev SV% Large w/ no EAC Change
StDev SV% Small w/ no EAC Change
StDev TSPI Large w/ no EAC Change
StDev TSPI Small w/ no EAC Change
StDev CV% Large w/ EAC Change
StDev CV% Small w/ EAC Change
CPI Change 1 Month Large w/ no EAC Change
CPI Change 1 Month Small w/ no EAC Change
SPI Change 1 Month Large w/ no EAC Change
SPI Change 1 Month Small w/ no EAC Change
SCI Change 1 Month Large w/ no EAC Change
SCI Change 1 Month Small w/ no EAC Change
SV% Change 1 Month Large w/ no EAC Change
SV% Change 1 Month Small w/ no EAC Change
CV% Change 1 Month Large w/ no EAC Change
CV% Change 1 Month Small w/ no EAC Change
TSPI Change 1 Month Large w/ no EAC Change
TSPI Change 1 Month Small w/ no EAC Change
TCPI Change 1 Month Large w/ no EAC Change
TCPI Change 1 Month Small w/ no EAC Change
CPI Change 2 Month Large w/ no EAC Change
CPI Change 2 Month Small w/ no EAC Change
SPI Change 2 Month Large w/ no EAC Change
SPI Change 2 Month Small w/ no EAC Change
SCI Change 2 Month Large w/ no EAC Change

SCI Change 2 Month Small w/ no EAC Change
SV% Change 2 Month Large w/ no EAC Change
SV% Change 2 Month Small w/ no EAC Change
TCPI Change 2 Month Large w/ no EAC Change
TCPI Change 2 Month Small w/ no EAC Change
TSPI Change 2 Month Large w/ no EAC Change
TSPI Change 2 Month Small w/ no EAC Change
Large% Difference Between ML and W w/ no EAC Change
Large% Difference Between ML and B w/ no EAC Change
Large% Difference Between B and W w/ no EAC Change
CPI Large w/ EAC Change
CPI Medium w/ EAC Change
CPI Small w/ EAC Change
SPI Large w/ EAC Change
SPI Medium w/ EAC Change
SPI Small w/ EAC Change
SCI Large w/ EAC Change
SCI Medium w/ EAC Change
SCI Small w/ EAC Change
TCPI Large w/ EAC Change
TCPI Medium w/ EAC Change
TCPI Small w/ EAC Change
TSPI Large w/ EAC Change
TSPI Medium w/ EAC Change
TSPI Small w/ EAC Change
SV% Large w/ EAC Change
SV% Medium w/ EAC Change
SV% Small w/ EAC Change
CV% Large w/ EAC Change
CV% Medium w/ EAC Change
CV% Small w/ EAC Change
StDev CPI Large w/ EAC Change
StDev CPI Small w/ EAC Change
StDev SPI Large w/ EAC Change
StDev SPI Small w/ EAC Change
StDev SCI Large w/ EAC Change
StDev SCI Small w/ EAC Change
StDev SV% Large w/ EAC Change
StDev SV% Small w/ EAC Change

StDev TSPI Large w/ EAC Change
StDev TSPI Small w/ EAC Change
StDev CV% Large w/ EAC Change
StDev CV% Small w/ EAC Change
CPI Change 1 Month Large w/ EAC Change
CPI Change 1 Month Small w/ EAC Change
SPI Change 1 Month Large w/ EAC Change
SPI Change 1 Month Small w/ EAC Change
SCI Change 1 Month Large w/ EAC Change
SCI Change 1 Month Small w/ EAC Change
SV% Change 1 Month Large w/ EAC Change
SV% Change 1 Month Small w/ EAC Change
CV% Change 1 Month Large w/ EAC Change
CV% Change 1 Month Small w/ EAC Change
TSPI Change 1 Month Large w/ EAC Change
TSPI Change 1 Month Small w/ EAC Change
TCPI Change 1 Month Large w/ EAC Change
TCPI Change 1 Month Small w/ EAC Change
CPI Change 2 Month Large w/ EAC Change
CPI Change 2 Month Small w/ EAC Change
SPI Change 2 Month Large w/ EAC Change
SPI Change 2 Month Small w/ EAC Change
SCI Change 2 Month Large w/ EAC Change
SCI Change 2 Month Small w/ EAC Change
SV% Change 2 Month Large w/ EAC Change
SV% Change 2 Month Small w/ EAC Change
TCPI Change 2 Month Large w/ EAC Change
TCPI Change 2 Month Small w/ EAC Change
TSPI Change 2 Month Large w/ EAC Change
TSPI Change 2 Month Small w/ EAC Change
Large% Difference Between ML and W w/ EAC Change
Large% Difference Between ML and B w/ EAC Change
Large% Difference Between B and W w/ EAC Change

Appendix E: List and Definition of Variables for Backwards Stepwise Regression

Variable	Description	Equation	Threshold
Intercept	The intercept for the overall equation	No equation	No
CPI	Ratio of budgeted work to actual work	$CPI = \frac{BCWP}{ACWP}$	No
SPI	Ratio of budgeted work to scheduled work	$SPI = \frac{BCWP}{BCWS}$	No
TCPI	Ratio of budgeted performance to actual performance	$TCPI = \frac{(BAC - BCWP)}{(EAC - ACWP)}$	No
TSPI	Ratio of the budgeted performance to schedule performance. This variable was only used in initial 100 variables and as part of threshold variables.	$TCPI = \frac{(BAC - BCWP)}{(BAC - BCWS)}$	
SCI	Cost ratio multiplied by schedule ratio	$SCI = CPI * SPI$	No
% Difference Between ML and B	The percentage difference between the contractor most likely EAC and best EAC	$\% \, Diff_{ML-B} = \frac{EAC_{ML} - EAC_B}{EAC_{ML}}$	No
% Difference Between W and B	The percentage difference between the contractor worst case EAC and best EAC	$\% \, Diff_{W-B} = \frac{EAC_W - EAC_B}{EAC_W}$	No

Standard Deviation CPI	A measure of the variability of the last three CPI's	$Stdev(CPI) = Stdev(CPI_t, CPI_{t-1}, CPI_{t-2})$	No
Standard Deviation SPI	A measure of the variability of the last three SPI's	$Stdev(SPI) = Stdev(SPI_t, SPI_{t-1}, SPI_{t-2})$	No
EAC Prediction CPI w/ no EAC Change	A gold card EAC prediction based on CPI that only turns on if the EAC has not changed by a threshold	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} < Threshold = \frac{\frac{ACWP+BAC-BCWP}{CPI}}{EAC_t} \\ \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold = 0 \end{cases}$	Yes
EAC Prediction Composite w/ no EAC Change	A gold card EAC prediction based on SCI that only turns on if the EAC has not changed by a threshold	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} < Threshold = \frac{\frac{ACWP+BAC-BCWP}{SCI}}{EAC_t} \\ \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold = 0 \end{cases}$	Yes
EAC Prediction CPI w/ EAC Change	A gold card EAC prediction based on CPI that only turns on if the EAC has changed by a threshold	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold = \frac{\frac{ACWP+BAC-BCWP}{CPI}}{EAC_t} \\ \frac{abs(EAC_1 - EAC_t)}{EAC_1} < Threshold = 0 \end{cases}$	Yes
EAC Prediction Composite w/ EAC Change	A gold card EAC prediction based on SCI that only turns on if the EAC has not changed by a threshold	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold = \frac{\frac{ACWP+BAC-BCWP}{SCI}}{EAC_t} \\ \frac{abs(EAC_1 - EAC_t)}{EAC_1} < Threshold = 0 \end{cases}$	Yes
CPI Large w/ EAC Change	The CPI exceeds some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and\ CPI > Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes

	throughout; however, threshold 2 is unique to this equation.		
CPI Small w/ EAC Change	The CPI is less than some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and\ CPI < Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes Yes
SPI Small w/ EAC Change	The SPI is less than some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and\ SPI < Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes
SCI Large w/ EAC Change	The SCI exceeds some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and\ SCI > Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes

TCPI Small w/ EAC Change	The TCPI is less than some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and\ TCPI < Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes
TSPI Large w/ EAC Change	The TSPI exceeds some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and\ TSPI > Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes
Standard Deviation CPI Large w/ EAC Change	The standard deviation of the CPI exceeds some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and\ stdev(CPI) > Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes

CPI Change 1 Month Large w/ EAC Change	The one-month change in CPI exceeds some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and \frac{CPI_t - CPI_{t-1}}{CPI_t} > Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes
SPI Change 1 Month Large w/ EAC Change	The one-month change in SPI exceeds some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and \frac{SPI_t - SPI_{t-1}}{SPI_t} > Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes
SCI Change 1 Month Small w/ EAC Change	The one-month change in SCI is less than some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and \frac{SCI_t - SCI_{t-1}}{SCI_t} < Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes

TSPI Change 1 Month Small w/ EAC Change	The one-month change in TSPI is less than some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and \frac{TSPI - TSPI_{t-1}}{TSPI_t} < Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes
CPI Change 2 Month Large w/ EAC Change	The two-month change in CPI exceeds some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and \frac{CPI_t - CPI_{t-2}}{CPI_t} > Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes
CPI Change 2 Month Small w/ EAC Change	The two-month change in CPI is less than some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and \frac{CPI_t - CPI_{t-2}}{CPI_t} < Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes

SPI Change 2 Month Large w/ EAC Change	The two-month change in SPI exceeds some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and \frac{SPI_t - SPI_{t-2}}{SPI_t} > Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes
Large% Difference Between B and W w/ EAC Change	The percent difference between best and worst contractor EAC exceeds some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and \frac{EAC_W - EAC_B}{EAC_W} > Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes
Small% Difference Between ML and W w/ no EAC Change	The percent difference between most likely and worst contractor EAC exceeds some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 and \frac{EAC_W - EAC_{ML}}{EAC_W} < Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes

	change is constant throughout; however, threshold 2 is unique to this equation.		
Small% Difference Between ML and B w/ no EAC Change	The percent difference between most likely and worst contractor EAC is smaller than some threshold and the most likely EAC has exceeded some threshold. The threshold for the EAC change is constant throughout; however, threshold 2 is unique to this equation.	$\begin{cases} \frac{abs(EAC_1 - EAC_t)}{EAC_1} > Threshold_1 \, and \, \frac{EAC_{ML} - EAC_B}{EAC_{ML}} < Threshold_2 = 1 \\ Else = 0 \end{cases}$	Yes

Appendix F: Code for Algorithm

```
Dim SSE As String, first_dynamic As String, count_beta As Integer, dynamic_var() As
Double, Sensitivity As Integer, Num_Var As Integer, solver_range As Variant,
remove_count As Integer, Solver_Count As Long, count_NA As Integer, numberx As
Long, remove_var_done As Integer
Dim Starting_point As Integer, Count_dynamic As Integer, move_dynamic_var As
Integer, Test_order_SSE As Double, best_order As Integer, Max_num_var As String,
Max_P As Double, Final_Optimize As Integer, place_SSE As Variant,
Final_Optimize_Percent As Double, DevSq As Double, Passing_P As Integer
Option Explicit
Sub get_inputs()
Dim count As Integer
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count
place_SSE = Range("A1").Offset(0, count_beta + 10).Address
SSE = place_SSE
remove_count = 0
first_dynamic = InputBox("What is the furthest left dynamic variable cell location", "
beta selection")
Max_num_var = InputBox("What is the greatest amount of variables you wish to have", "
Number of Variables")
Range("b6").Select
numberx = Range(Selection, Selection.End(xlDown)).count
Application.ScreenUpdating = False
Range("B2").Select
Range(Selection, Selection.End(xlToRight)).Select
Selection.Copy
Range("b2").Offset(5 + numberx, 0).Select
ActiveSheet.Paste
Application.CutCopyMode = False

Call Clear
Worksheets("MainCalculations").Select
Range(SSE).Offset(3, 0).Select
With Selection
   .Value = Now
End With
Range(first_dynamic).Offset(-1, -remove_count).Select
Count_dynamic = Range(Selection, Selection.End(xlToRight)).count
ReDim dynamic_var(1 To Count_dynamic, 1 To 5) As Double
Final_Optimize_Percent = 1
numberx = Range(Selection, Selection.End(xlDown)).count
Passing_P = 0
```

```
Call Determine_Dynamic_Start
Call Optimize_Dynamic
Call Calculations
Call Determine_Max_P
Call Clear

Do Until (Max_P < (0.05 / (count_beta - 1)) And Max_num_var >= (count_beta - 1)) Or
count_beta = 1
Passing_P = 0
   If Max_P > (0.05 / (count_beta - 1)) Then
      Call Remove_P_values
      remove_count = remove_count + 1
      Call Determine_Dynamic_Start
      Call Optimize_Dynamic
      Call Calculations
      Call Clear
      Call Determine_Max_P
   Else
Passing_P = 1
      Call Remove_Standard_Beta
      remove_count = remove_count + 1
      Call Determine_Dynamic_Start
      Call Optimize_Dynamic
      Call Calculations
      Call Clear
      Call Determine_Max_P
   End If
Loop

remove_var_done = 1
Call Determine_Dynamic_Order
Final_Optimize = 8
Final_Optimize_Percent = 0.000001
Call Optimize_Dynamic
Call Final_Calculations


Worksheets("MainCalculations").Select
Range(SSE).Offset(4, 0).Select
With Selection
   .Value = Now
End With
Range(SSE).Offset(5, 0) = Solver_Count
End Sub
Sub Determine_Dynamic_Order()
```

```
Dim Best_order_SSE As Double, count As Integer
Worksheets("MainCalculations").Select
Starting_point = 0
Range(first_dynamic).Offset(-1, -remove_count).Select
Count_dynamic = Range(Selection, Selection.End(xlToRight)).count

Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count
place_SSE = Range("A1").Offset(0, count_beta + 10).Address
SSE = place_SSE

count = 0

Do Until Starting_point = Count_dynamic - 1 - count_NA
   count = 0
   Best_order_SSE = 1 * 10 ^ 10
   move_dynamic_var = 0
   Do Until move_dynamic_var = Count_dynamic - Starting_point - count_NA
      If move_dynamic_var <> 0 Then
         Call move_dynamic
      End If
      Call Determine_Dynamic_Start
      Call Optimize_Dynamic
      If Test_order_SSE < Best_order_SSE Then
         Best_order_SSE = Test_order_SSE
         best_order = move_dynamic_var
      End If
      move_dynamic_var = move_dynamic_var + 1
      count = count + 1
   Loop
   move_dynamic_var = count - best_order - 1
      If move_dynamic_var <> 0 Then
         Range(first_dynamic).Offset(-1, Starting_point + move_dynamic_var -
remove_count).Select
         Range(Selection, Selection.End(xlDown)).Select
         Selection.Cut
         Range(first_dynamic).Offset(-1, Starting_point - remove_count).Select
         Selection.Insert Shift:=xlToRight
      End If
   Starting_point = Starting_point + 1
Loop


End Sub
```

```
Sub solver_solve()
Dim solver_range As Variant, result As Variant
SolverOptions MaxTime:=2000, Iterations:=20000, Precision:=0.005,
AssumeLinear:=False, StepThru:=False, Estimates:=1, Derivatives:=1, SearchOption:=1,
IntTolerance:=5, Scaling:=False, Convergence:=0.005, AssumeNonNeg:=False
Worksheets("StatisticalCalculations").Range("XFD1") = count_beta
solver_range = Worksheets("StatisticalCalculations").Range("XFD19")
SolverOk SetCell:=SSE, MaxMinVal:=2, ValueOf:="0",
ByChange:=Range(solver_range)
result = SolverSolve(True, True)
SolverSolve UserFinish:=True
Solver_Count = Solver_Count + 1



End Sub
Sub Determine_Dynamic_Start()
Dim count As Integer, Count_T As Integer, best_sse As Double, test_SSE As Double,
Count_Overall As Integer, test_dependents As Variant, no_error As Integer

Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count
place_SSE = Range("A1").Offset(0, count_beta + 10).Address
SSE = place_SSE
no_error = 0
count = 0




count = 0

If remove_var_done = 0 Then

NA_Finder_Start:
    Do Until count = Count_dynamic
    On Error GoTo Error_Handler_Start
    test_dependents = Worksheets("MainCalculations").Range(first_dynamic).Offset(0,
count - remove_count).Dependents
    count = count + 1
    Loop

count = 0
count_NA = 0
```

```
'Count NA dynamic variables
      Do Until count = Count_dynamic
      If Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = "NA" Then
         count_NA = count_NA + 1
      End If
      count = count + 1
      Loop

count = 0

'Move NA dynamic variables
      Do Until count = Count_dynamic - count_NA
      If Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = "NA" Then
         Worksheets("MainCalculations").Range(first_dynamic).Offset(-1, count -
remove_count).Select
         Range(Selection, Selection.End(xlDown)).Select
         Selection.Cut
         Worksheets("MainCalculations").Range(first_dynamic).Offset(-1, count -
remove_count).Select
         Selection.End(xlToRight).Select
         Selection.Offset(0, 1).Select
         Selection.Insert Shift:=xlToRight
         count = -1
      End If
      count = count + 1
      Loop
End If

count = 0

   'Enter low end of range for dynamic variables
      Do Until count = Count_dynamic
      dynamic_var(count + 1, 2) =
Worksheets("MainCalculations").Range(first_dynamic).Offset(1, count - remove_count)
      count = count + 1
      Loop

count = 0

   'Enter high end of range for dynamic variables
      Do Until count = Count_dynamic
```

```
        dynamic_var(count + 1, 3) =
Worksheets("MainCalculations").Range(first_dynamic).Offset(2, count - remove_count)
        count = count + 1
        Loop

count = 0

    'Enter range of dynamic variable
        Do Until count = Count_dynamic
        dynamic_var(count + 1, 4) = dynamic_var(count + 1, 3) - dynamic_var(count + 1, 2)
        count = count + 1
        Loop

'Find starting point for the dynamic variables
Do Until Count_Overall = 1
count = 0

    Do Until count = Count_dynamic - count_NA

        no_error = 0
        Count_T = 0
        If count = 0 And Count_Overall = 0 Then
            best_sse = 1 * 10 ^ 100
        End If

        Do Until Count_T = 20

        If Count_T = 0 Then
            dynamic_var(count + 1, 1) =
Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count - remove_count)
        End If

        Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = _
        dynamic_var(count + 1, 2) + (dynamic_var(count + 1, 4) / 20) * (Count_T)
        Call solver_solve
        test_SSE = Worksheets("MainCalculations").Range(SSE)

        If test_SSE < best_sse Then
            best_sse = test_SSE
            'Save Best SSE for starting point
            dynamic_var(count + 1, 5) = best_sse
            dynamic_var(count + 1, 1) = dynamic_var(count + 1, 2) + (dynamic_var(count +
1, 4) / 20) * (Count_T)
        End If
```

```
        Count_T = Count_T + 1
        no_error = 1

Error_Handler_Start:
   If no_error = 0 Then
        Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = "NA"
        On Error GoTo 0
        count = count + 1
        Resume NA_Finder_Start
   End If

        Loop
If no_error = 1 Then
   Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = dynamic_var(count + 1, 1)
   Call solver_solve
End If
   count = count + 1
   Loop
Count_Overall = Count_Overall + 1
Loop

End Sub
Sub Optimize_Dynamic()
Dim Change As Double, Count_Overall As Integer, count As Integer, Count_T As
Integer, Percent_Change As Double, old_value As Double, old_sse As Double,
Value_change_percent As Double, test_dependents As Variant
Dim value_change As Double, test_SSE As Double, best_sse, response1 As Variant,
start_value As Double, start_sse As Double, positive_direction_sse As Double,
negative_direction_sse As Double, no_error As Integer
Count_Overall = 0
Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count
place_SSE = Range("A1").Offset(0, count_beta + 10).Address
SSE = place_SSE
count = 0
no_error = 0
Range("b6").Select
numberx = Range(Selection, Selection.End(xlDown)).count

count = 0
```

```
If remove_var_done = 0 Then

'Find NA's
NA_Finder_Optimize:
	Do Until count = Count_dynamic
	On Error GoTo Error_Handler_Optimize
	test_dependents = Worksheets("MainCalculations").Range(first_dynamic).Offset(0,
count - remove_count).Dependents
	count = count + 1
	Loop

count = 0
count_NA = 0

'Count NA dynamic variables
	Do Until count = Count_dynamic
	If Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = "NA" Then
	   count_NA = count_NA + 1
	End If
	count = count + 1
	Loop

count = 0

'Move NA dynamic variables
	Do Until count = Count_dynamic - count_NA
	If Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = "NA" Then
	   Worksheets("MainCalculations").Range(first_dynamic).Offset(-1, count -
remove_count).Select
	   Range(Selection, Selection.End(xlDown)).Select
	   Selection.Cut
	   Worksheets("MainCalculations").Range(first_dynamic).Offset(-1, count -
remove_count).Select
	   Selection.End(xlToRight).Select
	   Selection.Offset(0, 1).Select
	   Selection.Insert Shift:=xlToRight
	   count = -1
	End If
	count = count + 1
	Loop

End If
```

```
count = 0

   'Enter low end of range for dynamic variables
      Do Until count = Count_dynamic
      dynamic_var(count + 1, 2) =
Worksheets("MainCalculations").Range(first_dynamic).Offset(1, count - remove_count)
      count = count + 1
      Loop

count = 0

   'Enter high end of range for dynamic variables
      Do Until count = Count_dynamic
      dynamic_var(count + 1, 3) =
Worksheets("MainCalculations").Range(first_dynamic).Offset(2, count - remove_count)
      count = count + 1
      Loop

count = 0

   'Enter range of dynamic variable
      Do Until count = Count_dynamic
      dynamic_var(count + 1, 4) = dynamic_var(count + 1, 3) - dynamic_var(count + 1, 2)
      count = count + 1
      Loop

count = 0

Do Until Count_Overall = 1 + Final_Optimize

count = 0

Do Until count = Count_dynamic - count_NA

      Count_T = 0
      Percent_Change = 0.001
      positive_direction_sse = 10 ^ 12
      negative_direction_sse = 10 ^ 12

   Do Until Percent_Change >= 0 And Percent_Change < 0.0001 *
Final_Optimize_Percent

If Count_T = 0 Then
```

```
    start_value = Worksheets("MainCalculations").Range(first_dynamic).Offset(0,
count - remove_count)
    start_sse = Worksheets("MainCalculations").Range(SSE)

        'save old coefficients if starting values better (hard for solver to optimize when
radically different)
        Range("b2").Select
        Range(Selection, Selection.End(xlToRight)).Copy
        Range("b2").Offset(5 + numberx, 0).Select
        ActiveSheet.Paste
        Application.CutCopyMode = False

    old_value = start_value
    old_sse = start_sse

'determine change direction
    Change = (dynamic_var(count + 1, 4) / 40) / (Count_Overall + 1)

    'Check to make sure positive change isn't outside positive range
    If old_value + Change < dynamic_var(count + 1, 3) Then
    Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = _
    old_value + Change
        Call solver_solve
        positive_direction_sse = Worksheets("MainCalculations").Range(SSE)
        Else
        positive_direction_sse = 10 ^ 12
    End If

    'Check to make sure positive change isn't outside positive range
    If old_value - Change > dynamic_var(count + 1, 2) Then
    Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = _
    old_value - Change
        Call solver_solve
        negative_direction_sse = Worksheets("MainCalculations").Range(SSE)
    Else
        negative_direction_sse = 10 ^ 12
    End If
    If negative_direction_sse < positive_direction_sse Then
        Change = Change * -1
        test_SSE = negative_direction_sse
    Else
        test_SSE = positive_direction_sse
    End If
```

```
    old_value = old_value + Change
    Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = old_value
    Call solver_solve
End If

If Count_T <> 0 Then


      old_sse = Worksheets("MainCalculations").Range(SSE)
      Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = _
      old_value + Change
      old_value = Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count
- remove_count)
      Call solver_solve
      test_SSE = Worksheets("MainCalculations").Range(SSE)
 End If

      Percent_Change = (old_sse - test_SSE) / old_sse

If Count_T = 200 Then
    Percent_Change = 0
End If

      If old_sse < test_SSE Then
         Change = Change * -0.5
      End If

      Count_T = Count_T + 1



If old_value + Change < dynamic_var(count + 1, 2) Or old_value + Change >
dynamic_var(count + 1, 3) Then
   Percent_Change = 0
End If

   Loop

   'save old value in array
    dynamic_var(count + 1, 1) = old_value

      'check to make sure not outside of lower range
```

```
      If Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) < dynamic_var(count + 1, 2) Then
         Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = dynamic_var(count + 1, 2)
         Call solver_solve
         best_sse = Worksheets("MainCalculations").Range(SSE)
         dynamic_var(count + 1, 1) =
Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count - remove_count)
         dynamic_var(count + 1, 5) = best_sse
         Percent_Change = 0
      End If

      'check to make sure not outside of upper range
      If Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) > dynamic_var(count + 1, 3) Then
         Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = dynamic_var(count + 1, 3)
         Call solver_solve
         test_SSE = Worksheets("MainCalculations").Range(SSE)
         dynamic_var(count + 1, 1) =
Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count - remove_count)
         dynamic_var(count + 1, 5) = best_sse
         Percent_Change = 0
      End If

      'check to make sure new sse is better than start sse
      If start_sse < test_SSE Then
         Range("b2").Offset(5 + numberx, 0).Select
         Range(Selection, Selection.End(xlToRight)).Copy
         Range("b2").Select
         ActiveSheet.Paste
         Application.CutCopyMode = False
         dynamic_var(count + 1, 1) = start_value
         start_sse = best_sse
         test_SSE = best_sse
         Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = start_value
         Call solver_solve
      End If

   Call solver_solve
   best_sse = Range(SSE)
   dynamic_var(count + 1, 5) = best_sse

   count = count + 1
```

```
        Loop
Call solver_solve
Test_order_SSE = Range(SSE)
Count_Overall = Count_Overall + 1
Loop

        no_error = 1

Error_Handler_Optimize:
   If no_error = 0 Then
        Worksheets("MainCalculations").Range(first_dynamic).Offset(0, count -
remove_count) = "NA"
        On Error GoTo 0
        count = count + 1
        Resume NA_Finder_Optimize
   End If

        Application.DisplayStatusBar = True
        If Passing_P = 0 Then
           Application.StatusBar = "Failing P's, " & count_beta & "Var's, MAPE " &
Round(Range(SSE).Offset(1, 0), 2)
           Else
           Application.StatusBar = "Failing P's, " & count_beta & "Var's, MAPE " &
Round(Range(SSE).Offset(1, 0), 2)
        End If
End Sub
Sub Calculations()
Dim title_end2, count_v As Long, count_h As Long, title_end3 As Variant, count_find
As Integer, SE_value As Double, result As Variant, count As Integer
Dim endval As String, endval2 As String, Title_end As String, tstat_value As Double,
xbar As Double, stdev_x As Double, Sum_stdBeta As Double, Error_range As Variant

Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count
place_SSE = Range("A1").Offset(0, count_beta + 10).Address
SSE = place_SSE
Range("b6").Select
numberx = Range(Selection, Selection.End(xlDown)).count

'Inverse Matrix Calculation
Worksheets("MainCalculations").Select
Worksheets("StatisticalCalculations").Select
Range("XFD1") = count_beta
```

```
Range("XFD3") = numberx
endval2 = Range("xfd4")
endval = Range("XFD2")
Title_end = Range("XFD5")
title_end2 = Range("xfd6")
Worksheets("StatisticalCalculations").Activate
Range("h2" & ":" & endval).Select
   Selection.FormulaArray = _
"=MINVERSE(MMULT(TRANSPOSE(MainCalculations!b6:" & endval2 &
"),MainCalculations!b6:" & endval2 & "))"

   'Title inverse matrix
   Range("h1:" & Title_end).Select
   With Selection
      .HorizontalAlignment = xlCenter
      .VerticalAlignment = xlBottom
      .WrapText = False
      .Orientation = 0
      .AddIndent = False
      .IndentLevel = 0
      .ShrinkToFit = False
      .ReadingOrder = xlContext
      .MergeCells = False
   End With
   Selection.Merge
   Range("h1:" & Title_end) = "X Inverse Matrix"

'Fill in chart
Worksheets("StatisticalCalculations").Range("c3") = count_beta - 1
Worksheets("StatisticalCalculations").Range("c4") = numberx - count_beta
Worksheets("StatisticalCalculations").Range("c5") = numberx - 1
Worksheets("StatisticalCalculations").Range("b4") =
Worksheets("MainCalculations").Range(SSE)
Worksheets("StatisticalCalculations").Range("b5") =
Application.WorksheetFunction.DevSq(Worksheets("MainCalculations").Range("A6:A"
& numberx + 5))
Worksheets("StatisticalCalculations").Range("g7") = numberx
Worksheets("StatisticalCalculations").Range("g8") = count_beta

'Variance-covariance matrix
count_v = 0
count_h = 0
Do Until count_h = count_beta
      Do Until count_v = count_beta
```

```
  Worksheets("StatisticalCalculations").Range("h2").Offset(count_beta + 2 + count_v,
count_h) = _
  Worksheets("StatisticalCalculations").Range("h2").Offset(count_v, count_h) *
Worksheets("StatisticalCalculations").Range("d4")
   count_v = count_v + 1
   Loop
   count_v = 0
   count_h = count_h + 1

Loop

'Title variance covariance matrix
   Range("h" & count_beta + 3 & ":" & title_end2).Select
   With Selection
      .HorizontalAlignment = xlCenter
      .VerticalAlignment = xlBottom
      .WrapText = False
      .Orientation = 0
      .AddIndent = False
      .IndentLevel = 0
      .ShrinkToFit = False
      .ReadingOrder = xlContext
      .MergeCells = False
   End With
   Selection.Merge
   Range("h" & count_beta + 3 & ":" & title_end2) = "Variance-Covariance Matrix"

'Correlation matrix

count_v = 0
count_h = 0
Worksheets("StatisticalCalculations").Range("XFD11") = 3
Worksheets("StatisticalCalculations").Range("XFD12") = 3
Dim static_start As String, static_end As String, dynamic_start As String, dynamic_end
As String
Do Until count_h = count_beta - 1



      Do Until count_v = count_beta - 1

static_start = Worksheets("StatisticalCalculations").Range("XFD7")
static_end = Worksheets("StatisticalCalculations").Range("XFD8")
dynamic_start = Worksheets("StatisticalCalculations").Range("XFD9")
dynamic_end = Worksheets("StatisticalCalculations").Range("XFD10")
```

```
        Worksheets("StatisticalCalculations").Range("h2").Offset(count_beta * 2 + 4 +
count_v, count_h) = _
        "=correl(MainCalculations!" & static_start & ":" & static_end &
",MainCalculations!" & dynamic_start & ":" & dynamic_end & ")"

         count_v = count_v + 1
         Worksheets("StatisticalCalculations").Range("XFD12") = 3 + count_v

    Loop

    count_v = 0
    count_h = count_h + 1
    Worksheets("StatisticalCalculations").Range("XFD11") = 3 + count_h
    Worksheets("StatisticalCalculations").Range("XFD12") = 3
Loop

'Title Correlation matrix

title_end3 = Worksheets("StatisticalCalculations").Range("XFD13")

    Range("h" & count_beta * 2 + 5 & ":" & title_end3).Select
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Selection.Merge
    Range("h" & count_beta * 2 + 5 & ":" & title_end3) = "Correlation Matrix"



'P-Value Variables

count = 0

Do Until count = count_beta
```

```
SE_value = (Worksheets("StatisticalCalculations").Range("h2").Offset(count_beta + 2 +
count, count)) ^ 0.5
tstat_value = Worksheets("MainCalculations").Range("B2").Offset(0, count) / SE_value
Worksheets("MainCalculations").Range("B5").Offset(0, count) = _
Application.WorksheetFunction.TDist(Abs(tstat_value),
Worksheets("StatisticalCalculations").Range("C4"), 1) * 2

count = count + 1

Loop

count = 0

'Calculate Standardized Beta's

   Sheets("MainCalculations").Select
   Cells.Select
   Range("C26").Activate
   Selection.Copy
   Sheets("StandardBeta").Select
   Range("A1").Select
   ActiveSheet.Paste
   Range("A1").Select
   Application.CutCopyMode = False
   Sheets("StandardBeta").Select
   Rows("1:5").Select
   Selection.ClearFormats
   Range("A1").Select
   count_h = 0

   Do Until count_h = count_beta - 1
      count_v = 0
      Worksheets("MainCalculations").Select
      Worksheets("MainCalculations").Range("C6").Offset(0, count_h).Select
      Range(Selection, Selection.End(xlDown)).Select
      xbar = Application.WorksheetFunction.Average(Selection)
      stdev_x = Application.WorksheetFunction.StDev(Selection)
      Worksheets("StandardBeta").Select

         Do Until count_v = numberx
         Range("C6").Offset(count_v, count_h) =
(Worksheets("MainCalculations").Range("C6").Offset(count_v, count_h) - xbar) /
stdev_x
         count_v = count_v + 1
         Loop
```

```
    count_h = count_h + 1
    Loop

Call solver_solve

count = 0

    Do Until count = count_beta - 1
    Range("C3").Offset(0, count) = Abs(Range("C2").Offset(0, count))
    count = count + 1
    Loop

Range("C3").Select
Range(Selection, Selection.End(xlToRight)).Select
Sum_stdBeta = Application.WorksheetFunction.Sum(Selection)

count = 0

    Do Until count = count_beta - 1
    Range("C4").Offset(0, count) = Abs(Range("C3").Offset(0, count)) / Sum_stdBeta
    count = count + 1
    Loop

    Range("C4").Select
    Range(Selection, Selection.End(xlToRight)).Select
    Selection.Copy
    Worksheets("MainCalculations").Select
    Worksheets("MainCalculations").Range("C4").Select
    ActiveSheet.Paste
    Application.CutCopyMode = False
    Range("A4") = "Standard Beta's"

Worksheets("MainCalculations").Select
End Sub
Sub Clear()
Worksheets("MainCalculations").Select
Range("b6").Select
numberx = Range(Selection, Selection.End(xlDown)).count

    Worksheets("StatisticalCalculations").Activate
    Range("H1:HZ703").Select
    Selection.Clear
    Range("G11").Select
    Range(Selection, Selection.End(xlToLeft)).Select
```

```
    Range("A11:G6119").Select
    Range("G11").Activate
    Selection.Clear
    Range("E8,E7,E6,C6,B5,B4,C3,C4,C5,F4,F5,G6,G7,G8,G10").Select
    Range("G8").Activate
    Selection.ClearContents
    Range("C20").Select
    Sheets("BPtest").Select
    Cells.Select
    Range("C20").Activate
    Selection.ClearContents
    Selection.ClearContents
    Sheets("StandardBeta").Select
    Cells.Select
    Range("L15").Activate
    Selection.Clear
    Range("A1").Select
Worksheets("MainCalculations").Select
Range("b2").Offset(5 + numberx, 0).Select
Range(Selection, Selection.End(xlToRight)).Select
Selection.Clear
End Sub
Sub move_dynamic()


    Range(first_dynamic).Offset(-1, Starting_point + move_dynamic_var -
remove_count).Select
    Range(Selection, Selection.End(xlDown)).Select
    Selection.Cut
    Range(first_dynamic).Offset(-1, Starting_point - remove_count).Select
    Selection.Insert Shift:=xlToRight
End Sub
Sub Remove_P_values()
Dim count As Integer, Max_Parent As Double, Min_None As Double, Min_Cross As
Double, Min_Power As Double
Dim test_parent As Double, test_none As Double, test_power As Double, test_cross As
Double, Max_P As Double, Temp As Variant, Mypos As Variant, strTemp As Variant
Dim left_word As String, right_word As String, word_length As Long, and_position As
Long, Min_Cross_Pos As Long, Min_Power_Pos As Long, Min_None_Pos As Long,
Max_Parent_Pos As Long
Dim Test_Text As String, left_word_test As String, right_word_test As String,
left_word_parent As Integer, right_word_parent As Integer, found_parent As Integer,
Power_parent As Integer

Worksheets("MainCalculations").Select
```

```
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count
count = 0
test_none = 1
test_power = 1
test_cross = 1
Min_None = 1
Min_Cross = 1
Min_Power = 1



'determine Min None stdBeta of failing p-values
Do Until count = count_beta - 1
      If UCase(Range("C5").Offset(-2, count)) = "NONE" And Range("C5").Offset(0,
count) > 0.05 / count_beta Then
         test_none = Range("C4").Offset(0, count)
            If test_none < Min_None Then
               Min_None = test_none
               Min_None_Pos = count
            End If
      End If
   count = count + 1
   Loop

count = 0

'determine Min Power stdBeta of failing p-values
   Do Until count = count_beta - 1
      If UCase(Range("C5").Offset(-2, count)) = "POWER" And Range("C5").Offset(0,
count) > 0.05 / count_beta Then
         test_power = Range("C4").Offset(0, count)
            If test_power < Min_Power Then
               Min_Power = test_power
               Min_Power_Pos = count
            End If
      End If
   count = count + 1
   Loop

count = 0

'determine Min Cross stdBeta of failing p-values
   Do Until count = count_beta - 1
```

```
      If UCase(Range("C5").Offset(-2, count)) = "CROSS" And Range("C5").Offset(0,
count) > 0.05 / count_beta Then
         test_cross = Range("C4").Offset(0, count)
            If test_cross < Min_Cross Then
               Min_Cross = test_cross
               Min_Cross_Pos = count
            End If
      End If
   count = count + 1
   Loop

count = 0

'Remove Max cross variable if it has a higher p value than max power and alpha crit
If Min_Cross <> 1 And Min_Cross < Min_Power And Min_Cross < Min_None Then

            'find crosses and remove parent label
            strTemp = Range("C1").Offset(0, Min_Cross_Pos)
            word_length = Len(strTemp)
            and_position = InStr(1, strTemp, "&", vbTextCompare)
            left_word = Left(strTemp, and_position - 1)
            right_word = Right(strTemp, word_length - and_position)

      Do Until count = count_beta

         Test_Text = Range("C1").Offset(0, count)

      If UCase(Range("C3").Offset(0, count)) = "CROSS" And UCase(Test_Text) <>
UCase(strTemp) Then

            word_length = Len(Test_Text)
            and_position = InStr(1, Test_Text, "&", vbTextCompare)
            left_word_test = Left(Test_Text, and_position - 1)
            right_word_test = Right(Test_Text, word_length - and_position)

               'Check to see if the to be removed first variable has any other crosses
               If left_word_test = left_word Or right_word_test = left_word Then
                  left_word_parent = 1
               End If

               'Check to see if the to be removed first variable has any other crosses
               If right_word_test = right_word Or left_word_test = right_word Then
                  right_word_parent = 1
               End If
```

```
        End If

            If UCase(Range("C3").Offset(0, count)) = "POWER" And UCase(Test_Text)
<> UCase(strTemp) Then

                'Check to see if the first word to be removed first variable has any other
Powers
                If Test_Text = left_word Then
                   left_word_parent = 1
                End If

                'Check to see if the second word to be removed first variable has any other
Powers
                If Test_Text = right_word Then
                   right_word_parent = 1
                End If

            End If

          count = count + 1

      Loop

      count = 0

      If right_word_parent = 0 Then
            Do Until found_parent = 1 Or count = count_beta - 1

                If UCase(Range("C3").Offset(0, count)) = "PARENT" And
Range("C1").Offset(0, count) = right_word Then
                      Range("C3").Offset(0, count) = "None"
                      found_parent = 1
                End If
              count = count + 1

            Loop
      End If

                count = 0
                found_parent = 0

      If left_word_parent = 0 Then
            Do Until found_parent = 1 Or count = count_beta - 1
```

```
                If UCase(Range("C3").Offset(0, count)) = "PARENT" And
Range("C1").Offset(0, count) = left_word Then
                    Range("C3").Offset(0, count) = "None"
                    found_parent = 1
                End If
            count = count + 1

        Loop
    End If

Worksheets("RemoveVariables").Select
Columns("A:A").Select
Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
Worksheets("MainCalculations").Select
Range("C1").Offset(0, Min_Cross_Pos).Select
Range(Selection, Selection.End(xlDown)).Select
   Range(Selection, Selection.End(xlDown)).Select
   Selection.Copy
   Sheets("RemoveVariables").Select
   Range("A1").Select
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
      :=False, Transpose:=False
Sheets("MainCalculations").Select
Application.CutCopyMode = False
Selection.Delete Shift:=xlToLeft

Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count
place_SSE = Range("A1").Offset(0, count_beta + 10).Address
SSE = place_SSE
Exit Sub

End If

'Remove max power variable if less than alpha crit
If Min_Power <> 1 And Min_Power < Min_None Then

            'find crosses and remove parent label
            strTemp = Range("C1").Offset(0, Min_Power_Pos)

    Do Until count = count_beta

        Test_Text = Range("C1").Offset(0, count)
```

```
        If UCase(Range("C3").Offset(0, count)) = "CROSS" And UCase(Test_Text) <>
UCase(strTemp) Then

                word_length = Len(Test_Text)
                and_position = InStr(1, Test_Text, "&", vbTextCompare)
                left_word_test = Left(Test_Text, and_position - 1)
                right_word_test = Right(Test_Text, word_length - and_position)

                   'Check to see if the to be removed variable has any other crosses
                   If left_word_test = strTemp Or right_word_test = strTemp Then
                      Power_parent = 1
                   End If

          End If

             If UCase(Range("C3").Offset(0, count)) = "POWER" And UCase(Test_Text)
<> UCase(strTemp) Then

                   'Check to see if the first word to be removed first variable has any other
Powers
                   If Test_Text = strTemp Then
                      Power_parent = 1
                   End If

             End If

            count = count + 1

      Loop

      count = 0

      If Power_parent = 0 Then
            Do Until found_parent = 1 Or count = count_beta - 1

                If UCase(Range("C3").Offset(0, count)) = "PARENT" And
Range("C1").Offset(0, count) = strTemp Then
                      Range("C3").Offset(0, count) = "None"
                      found_parent = 1
                End If
             count = count + 1

          Loop
      End If
```

```
Worksheets("RemoveVariables").Select
Columns("A:A").Select
Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
Worksheets("MainCalculations").Select
Range("C1").Offset(0, Min_Power_Pos).Select
Range(Selection, Selection.End(xlDown)).Select
   Range(Selection, Selection.End(xlDown)).Select
   Selection.Copy
   Sheets("RemoveVariables").Select
   Range("A1").Select
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
      :=False, Transpose:=False
Sheets("MainCalculations").Select
Application.CutCopyMode = False
Selection.Delete Shift:=xlToLeft

Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count
place_SSE = Range("A1").Offset(0, count_beta + 10).Address
SSE = place_SSE
Exit Sub

End If

'Remove Max None if Cross and Power are both less than alpha crit
Worksheets("RemoveVariables").Select
Columns("A:A").Select
Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
Worksheets("MainCalculations").Select
Range("C1").Offset(0, Min_None_Pos).Select
Range(Selection, Selection.End(xlDown)).Select
   Range(Selection, Selection.End(xlDown)).Select
   Selection.Copy
   Sheets("RemoveVariables").Select
   Range("A1").Select
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
      :=False, Transpose:=False
Sheets("MainCalculations").Select
Application.CutCopyMode = False
Selection.Delete Shift:=xlToLeft
```

```
Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count
place_SSE = Range("A1").Offset(0, count_beta + 10).Address
SSE = place_SSE


End Sub
Sub Remove_Standard_Beta()
Dim Min_stdBeta As Double, Test_stdBeta As Double, Min_stdBeta_Pos As Integer,
Min_stdBeta_Type As String, left_word_parent As Integer, right_word_parent As Integer
Dim count As Integer, strTemp As String, word_length As Integer, and_position As
Integer, left_word As String, right_word As String, Test_Text As String, left_word_test
As String
Dim right_word_test As String, found_parent As Integer, Power_parent As Integer

Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count
Min_stdBeta = 1

'Determine min standard beta
   Do Until count = count_beta - 1
      If UCase(Range("C3").Offset(0, count)) <> "PARENT" Then
         Test_stdBeta = Range("C4").Offset(0, count)
            If Test_stdBeta < Min_stdBeta Then
               Min_stdBeta = Test_stdBeta
               Min_stdBeta_Pos = count
               Min_stdBeta_Type = Range("C3").Offset(0, count)
            End If
      End If

   count = count + 1

   Loop

If UCase(Min_stdBeta_Type) = "CROSS" Then

count = 0

                'find crosses and remove parent label
                strTemp = Range("C1").Offset(0, Min_stdBeta_Pos)
                word_length = Len(strTemp)
                and_position = InStr(1, strTemp, "&", vbTextCompare)
                left_word = Left(strTemp, and_position - 1)
```

```
            right_word = Right(strTemp, word_length - and_position)

        Do Until count = count_beta

            Test_Text = Range("C1").Offset(0, count)

          If UCase(Range("C3").Offset(0, count)) = "CROSS" And UCase(Test_Text) <>
UCase(strTemp) Then

                word_length = Len(Test_Text)
                and_position = InStr(1, Test_Text, "&", vbTextCompare)
                left_word_test = Left(Test_Text, and_position - 1)
                right_word_test = Right(Test_Text, word_length - and_position)

                   'Check to see if the to be removed first variable has any other crosses
                   If left_word_test = left_word Or right_word_test = left_word Then
                      left_word_parent = 1
                   End If

                   'Check to see if the to be removed first variable has any other crosses
                   If right_word_test = right_word Or left_word_test = right_word Then
                      right_word_parent = 1
                   End If



            End If

              If UCase(Range("C3").Offset(0, count)) = "POWER" And UCase(Test_Text)
<> UCase(strTemp) Then

                   'Check to see if the first word to be removed first variable has any other
Powers
                   If Test_Text = left_word Then
                      left_word_parent = 1
                   End If

                   'Check to see if the second word to be removed first variable has any other
Powers
                   If Test_Text = right_word Then
                      right_word_parent = 1
                   End If

              End If
```

```
        count = count + 1

    Loop

    count = 0

    If right_word_parent = 0 Then
        Do Until found_parent = 1 Or count = count_beta - 1

            If UCase(Range("C3").Offset(0, count)) = "PARENT" And
Range("C1").Offset(0, count) = right_word Then
                Range("C3").Offset(0, count) = "None"
                found_parent = 1
            End If
        count = count + 1

        Loop
    End If

            count = 0
            found_parent = 0

    If left_word_parent = 0 Then
        Do Until found_parent = 1 Or count = count_beta - 1

            If UCase(Range("C3").Offset(0, count)) = "PARENT" And
Range("C1").Offset(0, count) = left_word Then
                Range("C3").Offset(0, count) = "None"
                found_parent = 1
            End If
        count = count + 1

        Loop
    End If

'Make new sheet to place viable model
If Max_num_var * 1.2 > count_beta Then
   Sheets.Add After:=Sheets(Sheets.count)
   Sheets(Sheets.count).Name = "Pass, " & count_beta & " var's"
   Sheets("MainCalculations").Select
   Cells.Select
   Application.Run "CB.CopyKeyPress"
   Worksheets("Pass, " & count_beta & " var's").Select
   ActiveSheet.Paste
   Sheets("StatisticalCalculations").Select
```

```
    Range("A1:G10").Select
    Application.Run "CB.CopyKeyPress"
    Sheets("Pass, " & count_beta & " var's").Select
    Range("XEX1").Select
    ActiveSheet.Paste
End If

Worksheets("RemoveVariables").Select
Columns("A:A").Select
Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
Worksheets("MainCalculations").Select
Range("C1").Offset(0, Min_stdBeta_Pos).Select
Range(Selection, Selection.End(xlDown)).Select
    Range(Selection, Selection.End(xlDown)).Select
    Selection.Copy
    Sheets("RemoveVariables").Select
    Range("A1").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
Sheets("MainCalculations").Select
Application.CutCopyMode = False
Selection.Delete Shift:=xlToLeft

Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count

place_SSE = Range("A1").Offset(0, count_beta + 10).Address
SSE = place_SSE
Exit Sub

End If

If UCase(Min_stdBeta_Type) = "POWER" Then

count = 0

                'find crosses and remove parent label
                strTemp = Range("C1").Offset(0, Min_stdBeta_Pos)

        Do Until count = count_beta

                Test_Text = Range("C1").Offset(0, count)
```

```
        If UCase(Range("C3").Offset(0, count)) = "CROSS" And UCase(Test_Text) <>
UCase(strTemp) Then

                word_length = Len(Test_Text)
                and_position = InStr(1, Test_Text, "&", vbTextCompare)
                count = count + 1
                left_word_test = Left(Test_Text, and_position - 1)
                right_word_test = Right(Test_Text, word_length - and_position)

                   'Check to see if the to be removed variable has any other crosses
                   If left_word_test = strTemp Or right_word_test = strTemp Then
                      Power_parent = 1
                   End If

        End If

           If UCase(Range("C3").Offset(0, count)) = "POWER" And UCase(Test_Text)
<> UCase(strTemp) Then

                   'Check to see if the first word to be removed first variable has any other
Powers
                   If Test_Text = strTemp Then
                      Power_parent = 1
                   End If

           End If

          count = count + 1

      Loop

      count = 0

      If Power_parent = 0 Then
              Do Until found_parent = 1 Or count = count_beta - 1

                 If UCase(Range("C3").Offset(0, count)) = "PARENT" And
Range("C1").Offset(0, count) = strTemp Then
                        Range("C3").Offset(0, count) = "None"
                        found_parent = 1
                 End If
              count = count + 1

           Loop
      End If
```

```
'Make new sheet to place viable model
If Max_num_var * 1.2 > count_beta Then
   Sheets.Add After:=Sheets(Sheets.count)
   Sheets(Sheets.count).Name = "Pass, " & count_beta & " var's"
   Sheets("MainCalculations").Select
   Cells.Select
   Application.Run "CB.CopyKeyPress"
   Worksheets("Pass, " & count_beta & " var's").Select
   ActiveSheet.Paste
   Sheets("StatisticalCalculations").Select
   Range("A1:G10").Select
   Application.Run "CB.CopyKeyPress"
   Sheets("Pass, " & count_beta & " var's").Select
   Range("XEX1").Select
   ActiveSheet.Paste
End If

Worksheets("RemoveVariables").Select
Columns("A:A").Select
Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
Worksheets("MainCalculations").Select
Range("C1").Offset(0, Min_stdBeta_Pos).Select
Range(Selection, Selection.End(xlDown)).Select
   Range(Selection, Selection.End(xlDown)).Select
   Selection.Copy
   Sheets("RemoveVariables").Select
   Range("A1").Select
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
      :=False, Transpose:=False
Sheets("MainCalculations").Select
Application.CutCopyMode = False
Selection.Delete Shift:=xlToLeft

Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count

place_SSE = Range("A1").Offset(0, count_beta + 10).Address
SSE = place_SSE
Exit Sub

End If

If UCase(Min_stdBeta_Type) = "NONE" Then
```

```
'Make new sheet to place viable model
If Max_num_var * 1.2 > count_beta Then
   Sheets.Add After:=Sheets(Sheets.count)
   Sheets(Sheets.count).Name = "Pass, " & count_beta & " var's"
   Sheets("MainCalculations").Select
   Cells.Select
   Application.Run "CB.CopyKeyPress"
   Worksheets("Pass, " & count_beta & " var's").Select
   ActiveSheet.Paste
   Sheets("StatisticalCalculations").Select
   Range("A1:G10").Select
   Application.Run "CB.CopyKeyPress"
   Sheets("Pass, " & count_beta & " var's").Select
   Range("XEX1").Select
   ActiveSheet.Paste
End If

Worksheets("RemoveVariables").Select
Columns("A:A").Select
Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove
Worksheets("MainCalculations").Select
Range("C1").Offset(0, Min_stdBeta_Pos).Select
Range(Selection, Selection.End(xlDown)).Select
   Range(Selection, Selection.End(xlDown)).Select
   Selection.Copy
   Sheets("RemoveVariables").Select
   Range("A1").Select
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
      :=False, Transpose:=False
Sheets("MainCalculations").Select
Application.CutCopyMode = False
Selection.Delete Shift:=xlToLeft

Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count

place_SSE = Range("A1").Offset(0, count_beta + 10).Address
SSE = place_SSE
End If

End Sub
Sub Determine_Max_P()
Dim test_p As Double, count As Integer
```

```
Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count
Worksheets("MainCalculations").Select
Max_P = 0
   'find max p value of non parent variables
   Do Until count = count_beta - 1
      If UCase(Range("C5").Offset(-2, count)) <> "PARENT" Then
         test_p = Range("C5").Offset(0, count)
               If test_p > Max_P Then
                  Max_P = test_p
               End If
      End If
      count = count + 1
   Loop


End Sub
Sub Final_Calculations()
Dim numberx As Long, title_end2, count_v As Long, count_h As Long, title_end3 As
Variant, count_find As Integer, SE_value As Double, result As Variant
Dim endval As String, endval2 As String, Title_end As String, tstat_value As Double,
xbar As Double, stdev_x As Double, Sum_stdBeta As Double, Error_range As Variant

Worksheets("MainCalculations").Select
Range("B2").Select
count_beta = Range(Selection, Selection.End(xlToRight)).count
place_SSE = Range("A1").Offset(0, count_beta + 10).Address
SSE = place_SSE
Range("b6").Select
numberx = Range(Selection, Selection.End(xlDown)).count

'Inverse Matrix Calculation
Worksheets("MainCalculations").Select
Range("c6").Select
numberx = Range(Selection, Selection.End(xlDown)).count
Worksheets("StatisticalCalculations").Select
Range("XFD1") = count_beta
Range("XFD3") = numberx
endval2 = Range("xfd4")
endval = Range("XFD2")
Title_end = Range("XFD5")
title_end2 = Range("xfd6")
Worksheets("StatisticalCalculations").Activate
Range("h2" & ":" & endval).Select
```

```
    Selection.FormulaArray = _
"=MINVERSE(MMULT(TRANSPOSE(MainCalculations!b6:" & endval2 &
"),MainCalculations!b6:" & endval2 & "))"

    'Title inverse matrix
    Range("h1:" & Title_end).Select
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Selection.Merge
    Range("h1:" & Title_end) = "X Inverse Matrix"

'Fill in chart
Worksheets("StatisticalCalculations").Range("c3") = count_beta - 1
Worksheets("StatisticalCalculations").Range("c4") = numberx - count_beta
Worksheets("StatisticalCalculations").Range("c5") = numberx - 1
Worksheets("StatisticalCalculations").Range("b4") =
Worksheets("MainCalculations").Range(SSE)
Worksheets("StatisticalCalculations").Range("b5") =
Application.WorksheetFunction.DevSq(Worksheets("MainCalculations").Range("A6:A"
& numberx + 5))
Worksheets("StatisticalCalculations").Range("g7") = numberx
Worksheets("StatisticalCalculations").Range("g8") = count_beta

'Variance-covariance matrix
count_v = 0
count_h = 0
Do Until count_h = count_beta
        Do Until count_v = count_beta
   Worksheets("StatisticalCalculations").Range("h2").Offset(count_beta + 2 + count_v,
count_h) = _
   Worksheets("StatisticalCalculations").Range("h2").Offset(count_v, count_h) *
Worksheets("StatisticalCalculations").Range("d4")
    count_v = count_v + 1
    Loop
    count_v = 0
    count_h = count_h + 1
```

```
Loop

'Title variance covariance matrix
   Range("h" & count_beta + 3 & ":" & title_end2).Select
   With Selection
      .HorizontalAlignment = xlCenter
      .VerticalAlignment = xlBottom
      .WrapText = False
      .Orientation = 0
      .AddIndent = False
      .IndentLevel = 0
      .ShrinkToFit = False
      .ReadingOrder = xlContext
      .MergeCells = False
   End With
   Selection.Merge
   Range("h" & count_beta + 3 & ":" & title_end2) = "Variance-Covariance Matrix"

'Correlation matrix

count_v = 0
count_h = 0
Worksheets("StatisticalCalculations").Range("XFD11") = 3
Worksheets("StatisticalCalculations").Range("XFD12") = 3
Dim static_start As String, static_end As String, dynamic_start As String, dynamic_end
As String
Do Until count_h = count_beta - 1



      Do Until count_v = count_beta - 1

static_start = Worksheets("StatisticalCalculations").Range("XFD7")
static_end = Worksheets("StatisticalCalculations").Range("XFD8")
dynamic_start = Worksheets("StatisticalCalculations").Range("XFD9")
dynamic_end = Worksheets("StatisticalCalculations").Range("XFD10")

      Worksheets("StatisticalCalculations").Range("h2").Offset(count_beta * 2 + 4 +
count_v, count_h) = _
      "=correl(MainCalculations!" & static_start & ":" & static_end &
",MainCalculations!" & dynamic_start & ":" & dynamic_end & ")"

       count_v = count_v + 1
       Worksheets("StatisticalCalculations").Range("XFD12") = 3 + count_v
```

```
    Loop

    count_v = 0
    count_h = count_h + 1
    Worksheets("StatisticalCalculations").Range("XFD11") = 3 + count_h
    Worksheets("StatisticalCalculations").Range("XFD12") = 3
Loop

'Title Correlation matrix

title_end3 = Worksheets("StatisticalCalculations").Range("XFD13")

    Range("h" & count_beta * 2 + 5 & ":" & title_end3).Select
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Selection.Merge
    Range("h" & count_beta * 2 + 5 & ":" & title_end3) = "Correlation Matrix"


'calculate durbin watson

Dim error_cell As String, count As Integer, Count_error As Long

    'determine which cell has error
        Do Until UCase(error_cell) = "ERROR"
            error_cell = Worksheets("MainCalculations").Range("A1").Offset(0, count_find)
                count_find = count_find + 1
        Loop
    Do Until Count_error = numberx - 1
        Worksheets("StatisticalCalculations").Range("b12").Offset(Count_error, 0) =
(Worksheets("MainCalculations").Range("A1").Offset(5 + Count_error, count_find - 1) -
_
        Worksheets("MainCalculations").Range("A1").Offset(5 + Count_error + 1,
count_find - 1)) ^ 2
        Count_error = Count_error + 1
```

```
    Loop



'Leverage
Dim Var_moving As String, X_inv As String

count = 0

Do Until count = numberx

    Worksheets("StatisticalCalculations").Range("XFD14") = count
    Var_moving = Worksheets("StatisticalCalculations").Range("XFD15")
    X_inv = Worksheets("StatisticalCalculations").Range("XFD16")

    Worksheets("StatisticalCalculations").Range("c11").Offset(count, 0) =
"=Sumproduct(Mmult(" & "MainCalculations!" & Var_moving & "," &
"StatisticalCalculations!" & X_inv & ")," & "MainCalculations!" & Var_moving & ")"

count = count + 1

Loop

'Studentized Residuals

count = 0

Do Until count = numberx

    Worksheets("StatisticalCalculations").Range("D11").Offset(count, 0) =
Worksheets("MainCalculations").Range("A1").Offset(5 + count, count_find - 1) / _
    (Worksheets("StatisticalCalculations").Range("D4") * (1 -
Worksheets("StatisticalCalculations").Range("C11").Offset(count, 0))) ^ 0.5

    count = count + 1


Loop

Worksheets("StatisticalCalculations").Range("D11").Select
    Range(Selection, Selection.End(xlDown)).Select
    ActiveWorkbook.Worksheets("StatisticalCalculations").sort.SortFields.Clear
    ActiveWorkbook.Worksheets("StatisticalCalculations").sort.SortFields.Add
Key:=Range("D11"), _
        SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
```

```
    With ActiveWorkbook.Worksheets("StatisticalCalculations").sort
        .SetRange Range("D11:D" & numberx + 10)
        .Header = xlNo
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
    Range("A1").Select

'Cooks Distance
count = 0

Do Until count = numberx

    Worksheets("StatisticalCalculations").Range("F11").Offset(count, 0) =
(Worksheets("MainCalculations").Range("A1").Offset(5 + count, count_find - 1)) ^ 2 / _
    (Worksheets("StatisticalCalculations").Range("G8") *
Worksheets("StatisticalCalculations").Range("D4")) * _
    (Worksheets("StatisticalCalculations").Range("C11").Offset(count, 0) / (1 -
Worksheets("StatisticalCalculations").Range("C11").Offset(count, 0)) ^ 2)

    count = count + 1


Loop
    Range("G10") = "=Max(F11:F" & numberx + 10 & ")"


'P-Value Variables

count = 0

Do Until count = count_beta
SE_value = (Worksheets("StatisticalCalculations").Range("h2").Offset(count_beta + 2 +
count, count)) ^ 0.5
tstat_value = Worksheets("MainCalculations").Range("B2").Offset(0, count) / SE_value
Worksheets("MainCalculations").Range("B5").Offset(0, count) = _
Application.WorksheetFunction.TDist(Abs(tstat_value),
Worksheets("StatisticalCalculations").Range("C4"), 1) * 2

count = count + 1

Loop
```

```
count = 0

'Calculate Standardized Beta's

   Sheets("MainCalculations").Select
   Cells.Select
   Range("C26").Activate
   Selection.Copy
   Sheets("StandardBeta").Select
   Range("A1").Select
   ActiveSheet.Paste
   Range("A1").Select
   Application.CutCopyMode = False
   Sheets("StandardBeta").Select
   Rows("1:5").Select
   Selection.ClearFormats
   Range("A1").Select
   count_h = 0

   Do Until count_h = count_beta - 1
      count_v = 0
      Worksheets("MainCalculations").Select
      Worksheets("MainCalculations").Range("C6").Offset(0, count_h).Select
      Range(Selection, Selection.End(xlDown)).Select
      xbar = Application.WorksheetFunction.Average(Selection)
      stdev_x = Application.WorksheetFunction.StDev(Selection)
      Worksheets("StandardBeta").Select

         Do Until count_v = numberx
         Range("C6").Offset(count_v, count_h) =
(Worksheets("MainCalculations").Range("C6").Offset(count_v, count_h) - xbar) /
stdev_x
         count_v = count_v + 1
         Loop

   count_h = count_h + 1
   Loop

Call solver_solve

count = 0

   Do Until count = count_beta - 1
   Range("C3").Offset(0, count) = Abs(Range("C2").Offset(0, count))
   count = count + 1
```

```
    Loop

Range("C3").Select
Range(Selection, Selection.End(xlToRight)).Select
Sum_stdBeta = Application.WorksheetFunction.Sum(Selection)

count = 0

    Do Until count = count_beta - 1
    Range("C4").Offset(0, count) = Abs(Range("C3").Offset(0, count)) / Sum_stdBeta
    count = count + 1
    Loop

    Range("C4").Select
    Range(Selection, Selection.End(xlToRight)).Select
    Selection.Copy
    Worksheets("MainCalculations").Select
    Worksheets("MainCalculations").Range("C4").Select
    ActiveSheet.Paste
    Application.CutCopyMode = False
    Range("A4") = "Standard Beta's"


'run solver against squared residuals

    'copy and paste
    Range("B14").Select
    Sheets("MainCalculations").Select
    Cells.Select
    Selection.Copy
    Sheets("BPtest").Select
    Range("A1").Select
    ActiveSheet.Paste
    Sheets("MainCalculations").Select
    Range("D87").Select
    ActiveWindow.SmallScroll Down:=-48
    Application.CutCopyMode = False
    Range("A1").Offset(5, count_find).Select
    Range(Selection, Selection.End(xlDown)).Select
    Selection.Copy
    Sheets("BPtest").Select
    Range("A6").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Range("A1").Select
```

```
Worksheets("BPtest").Activate

SolverOptions Precision:=0.001
solver_range = Worksheets("StatisticalCalculations").Range("XFD19")
SolverOk SetCell:=SSE, MaxMinVal:=2, ValueOf:="0",
ByChange:=Range(solver_range)
result = SolverSolve(True, True)
SolverSolve UserFinish:=True

'Get SSR(resid) and R-sq (resid)
Worksheets("StatisticalCalculations").Range("f5") =
Application.WorksheetFunction.DevSq(Worksheets("BPtest").Range("A6:A" & numberx
+ 5)) _
- Worksheets("BPtest").Range(SSE)
Worksheets("StatisticalCalculations").Range("f4") =
Worksheets("StatisticalCalculations").Range("f5") / _
Application.WorksheetFunction.DevSq(Worksheets("BPtest").Range("A6:A" & numberx
+ 5))

'Jaque-Berra test
Worksheets("StatisticalCalculations").Range("xfd17") = count_find
Error_range = Worksheets("StatisticalCalculations").Range("xfd18")
Worksheets("StatisticalCalculations").Range("g6") = "=JB_test(MainCalculations!" &
Error_range & ")"

'Durbin-Watson test stat
Worksheets("StatisticalCalculations").Range("E8") = "=sum(B12:B" & numberx + 10 &
")/B4"

'AD Test

Worksheets("StatisticalCalculations").Range("E6") = "=AD(MainCalculations!" &
Error_range & ")"

'KS test
Worksheets("StatisticalCalculations").Range("E7") =
"=KS(StatisticalCalculations!XFA1:XFA10000,D11:D" & numberx + 10 & ")"


Worksheets("MainCalculations").Select
End Sub
```

```
Public Function JB_test(Normal_test_vals As Range) As Variant
```

```
Dim Norm_test() As Double

Norm_testR = Normal_test_vals.Value2
n = UBound(Norm_testR, 1)

ReDim Norm_test(1 To n)

count = 1

Do Until count = n + 1
Norm_test(count) = Norm_testR(count, 1)
count = count + 1
Loop

norm_test_ave = Application.WorksheetFunction.Average(Norm_test)

count = 1

Do Until count = n + 1

s_top = s_top + (Norm_test(count) - norm_test_ave) ^ 3
s_bottom = s_bottom + (Norm_test(count) - norm_test_ave) ^ 2
k_top = k_top + (Norm_test(count) - norm_test_ave) ^ 4
k_bottom = k_bottom + (Norm_test(count) - norm_test_ave) ^ 2
count = count + 1
Loop

S = (s_top * 1 / n) / ((s_bottom * 1 / n) ^ (3 / 2))
K = (1 / n * k_top) / ((1 / n * k_bottom) ^ 2) - 3

JB_test = n / 6 * (S ^ 2 + 1 / 4 * K ^ 2)

End Function

Public Function KS(Actual As Range, Test As Range) As Variant
Application.Volatile

Dim ProbAct() As Double
Dim ProbSim() As Double
Dim nA As Integer
Dim nS As Integer
Dim nLarge As Integer
Dim Act() As Double
Dim Sim() As Double
```

```
ActR = Actual.Value2
SimR = Test.Value2


nA = UBound(ActR, 1)
nS = UBound(SimR, 1)
ReDim ProbAct(1 To nA)
ReDim ProbSim(1 To nS)
ReDim Act(1 To nA)
ReDim Sim(1 To nS)

If nA > nS Then
   nLarge = nA
   Else
   nLarge = nS
End If

Do Until nA = count
   count = count + 1
   Act(count) = ActR(count, 1)
Loop

count = 0

Do Until nS = count
   count = count + 1
   Sim(count) = SimR(count, 1)
Loop

count = 0

Do Until count = nLarge + 1
    count = count + 1
      If count <= nA Then
         ProbAct(count) = (count) / (nA)
      End If

   If count <= nS Then
   ProbSim(count) = (count) / (nS)
   End If

Loop

Dim First As Integer
```

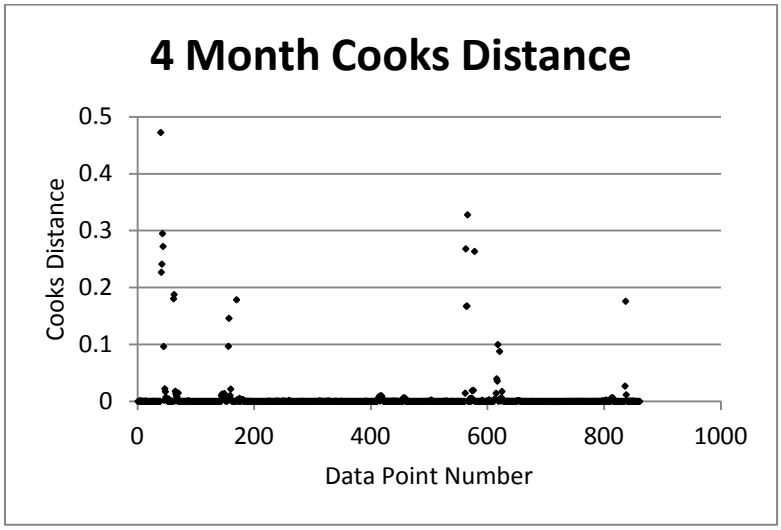
```
Dim Last As Integer
Dim i As Integer
Dim j As Integer
Dim Temp As String

   First = LBound(Sim)
   Last = UBound(Sim)
   For i = First To Last - 1
      For j = i + 1 To Last
         If Sim(i) > Sim(j) Then
            Temp = Sim(j)
            Sim(j) = Sim(i)
            Sim(i) = Temp
         End If
      Next j
   Next i

      For i = 1 To UBound(Sim)
         Debug.Print Sim(i)
      Next i

i = 0
j = 0

   First = LBound(Act)
   Last = UBound(Act)
   For i = First To Last - 1
      For j = i + 1 To Last
         If Act(i) > Act(j) Then
            Temp = Act(j)
            Act(j) = Act(i)
            Act(i) = Temp
         End If
      Next j
   Next i

      For i = 1 To UBound(Act)
         Debug.Print Act(i)
      Next i

count = 0
y = 1

Do Until x = nS
x = x + 1
```

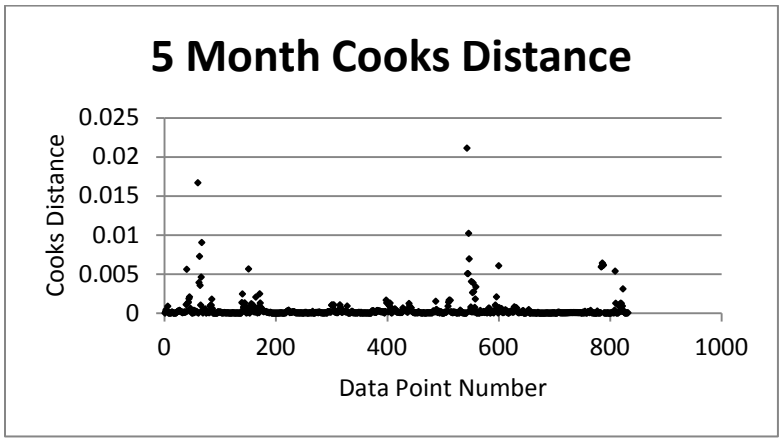
```
    If Sim(x) < Act(1) Then
    Do Until Sim(x) > Act(1)
       D = ProbSim(x) - 0
       If D > DFinal Then
          DFinal = D
       End If
       x = x + 1
    Loop
    End If

    Do Until Sim(x) > Act(y) And Sim(x) < Act(y + 1) Or Sim(x) > Act(nA)
       y = y + 1
    Loop

    If Abs(Sim(x) - Act(y)) < Abs(Sim(x) - Act(y + 1)) Then
       D = Abs(ProbSim(x) - ProbAct(y))
             If D > DFinal Then
                DFinal = D
             End If
    Else
     D = Abs(ProbSim(x) - ProbAct(y + 1))
             If D > DFinal Then
                DFinal = D
             End If
     End If



Loop
KS = DFinal

End Function

Public Function AD(Normal_test_vals As Range) As Variant

Dim Norm_test() As Double, Prob() As Double
Dim S_ad As Double

Norm_testR = Normal_test_vals.Value2
n = UBound(Norm_testR, 1)

ReDim Norm_test(1 To n)
ReDim Prob(1 To n)

count = 1
```

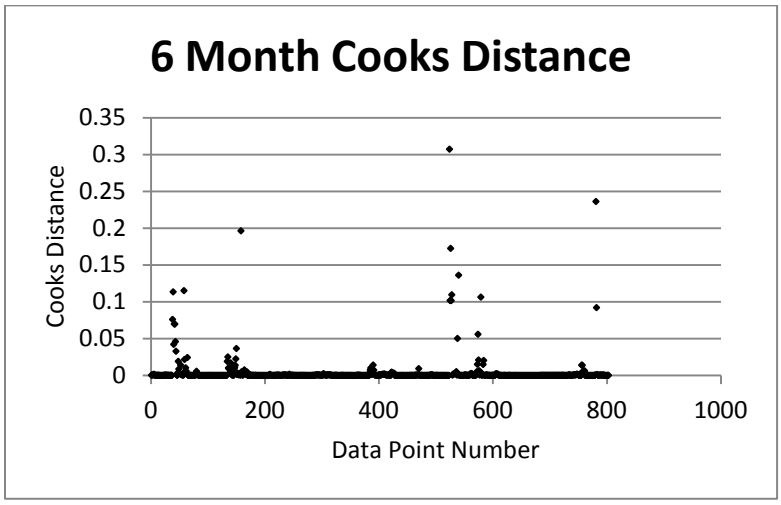
```
Do Until count = n + 1
Norm_test(count) = Norm_testR(count, 1)
count = count + 1
Loop

norm_test_ave = Application.WorksheetFunction.Average(Norm_test)
norm_test_std = Application.WorksheetFunction.StDev(Norm_test)


Dim First As Integer
Dim Last As Integer
Dim i As Integer
Dim j As Integer
Dim Temp As String

   First = LBound(Norm_test)
   Last = UBound(Norm_test)
   For i = First To Last - 1
      For j = i + 1 To Last
         If Norm_test(i) > Norm_test(j) Then
            Temp = Norm_test(j)
            Norm_test(j) = Norm_test(i)
            Norm_test(i) = Temp
         End If
      Next j
   Next i

      For i = 1 To UBound(Norm_test)
         Debug.Print Norm_test(i)
      Next i

count = 1

Do Until count = n + 1
         Prob(count) = Application.WorksheetFunction.NormDist(Norm_test(count),
norm_test_ave, norm_test_std, True)
            count = count + 1
Loop

count = 1

Do Until count = n + 1
   S_ad = S_ad + ((2 * count - 1) / n) * (Log(Prob(count)) + Log(1 - Prob(n - count + 1)))
   count = count + 1
```

```
Loop

A_sq = -n - S_ad

AD = A_sq * (1 + 0.75 / n + 2.25 / n ^ 2)

End Function
```

Appendix G: Cooks Distance Plots







Bibliography

Andrew, D. P., Pedersen, P. M., & McEvoy, C. D. (2011). *Research methods and design in sport management*. Champaign IL: Human Kinetics.

Arena, M. V., Leonard, R. S., Murray, S. E., & Younossi, O. (1994). Historical Cost Growth of Completed Weapon System Programs. *RAND Corporation, 1*, 1-47.

Armstrong, J. S. (2002). *Principles of forecasting a handbook for researchers and practitioners*. New York: Kluwer Academic Publishers.

Bart, J., & Fligner, M. A. (1998). *Sampling and statistical methods for behavioral ecologists*. Cambridge: Cambridge University Press.

Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge UK: Cambridge University Press.

Chatfield, C. (2001). *Time-series forecasting*. Boca Raton: Chapman & Hall/CRC.

Chen, D., & Batson, R. G. (2010). *Applied integer programming: modeling and solution*. Hoboken NJ: Wiley.

Christensen, D. S., & Payne, K. (1992). CPI Stability Fact or Fiction. *Journal of Parametrics, 10*, 27-40.

Cook, D. R. (1977). Detection of Influential Observation in Linear Regression. *Technometrics, 19*(1), 15-18.

Department of Commerce. (n.d.). *Department of Commerce*. Retrieved January 20, 2012, from http://www.commerce.gov/

Erdogmus, H. (2010). Tracking Progress through Earned Value. *IEEE Computer Society, 0740-7459*(10), 2-7.

Floudas C. A. (2001). *Encyclopedia of optimization*. Dordrecht: Kluwer Academic.

Gross, J. (2003). *Linear regression*. Berlin: Springer.

Gutin, G., & Punnen, A. P. (2002). *The traveling salesman problem and its variations*. Dordrecht: Kluwer Academic Publishers.

Harshbarger, R. J., & Reynolds, J. J. (2008). *Mathematical Applications: for the Management, Life and Social Sciences* (9th ed.). Boston: Houghton Mifflin Company.

Air Force Materiel Command. (1994, April 4). Guide to Analysis of Contractor Cost Data. *Financial Management*, *1*, 1-103.

Keaton, C. G., White, E. D., & Unger, E. J. (2011). Using Earned Value Data to Detect Potential Problems in Acquisition Contracts. *Journal of Cost Analysis and Parametrics*, *4*(2), 148-159 .

Kerzner, H. (2009). *Project management a systems approach to planning, scheduling, and controlling* (10th ed.). New York: John Wiley.

Lorenz, A., Bosch, H., & Küttler, K. (2011). Implementation of Earned Value Management tools in the Wendel stein 7-X Project. *IEEEINPSS 24th Symposium on Fusion Engineering*, *S04C*(4), 1-4.

Makridakis, S., Hyndman, R. J., & Wheelwright, S. C. (1998). *Forecasting: methods and applications* (3. ed.). New York NY: Wiley.

Mitchell, J. E., & Borchers, B. (2000). Chapter 14. *High performance optimization* (pp. 349-366). Dordrecht: Kluwer Academic Publishers.

Montgomery, D. C., Jennings, C. L., & Kulahci, M. (2008). *Introduction to time series*

analysis and forecasting. Hoboken N.J.: Wiley-Interscience.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2011). *Introduction to linear regression analysis* (5th ed.). Oxford: Wiley-Blackwell.

Morin, J. M. (2010). Achieving Acquisition Excellence in the Air Force: A Financial Management Perspective. *Armed Forces Comptroller , 55*(2), 8-12.

Nelder, J., & Mead, R. (1965). A Simplex Method for function minimization. *The Computer Journal, 7*(4), 308-313.

Rachev, S. T. (2007). *Financial econometrics: from basics to advanced modeling techniques*. Hoboken NJ: Wiley.

Rencher, A. C. (2002). *Methods of multivariate analysis* (2nd ed.). New York: J. Wiley.

Shumway, R. H., & Stoffer, D. S. (2000). *Time series analysis and its applications*. New York: Springer.

Younossi, O., Arena, M. V., Leonard, R. S., Roll, C. R., Jain, A., & Sollinger, J. M. (2007). *Is weapon system cost growth increasing?: a quantitative assessment of completed and ongoing programs.* . Santa Monica: RAND.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
Grad Date, Ex: 29 Mar 2012	Master's Thesis	Ex: 04 Aug 2010 - 29 Mar 2012

4. TITLE AND SUBTITLE
Using Predictive Analytics to Detect Major Problems in Department of Defense Acquisition Programs

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Dowling, Austin W. First Lieutenant, USAF

5d. PROJECT NUMBER
JON 12C135

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCA/ENC/12-03

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Mr. Steve Miller
Office of the Secretary of Defense, Cost Assessment and Program Evaluation
Room BE-829 1800 Defense Pentagon
Washington, DC 20301-1800
(703) 697-5056

10. SPONSOR/MONITOR'S ACRONYM(S)
OSD/CAPE

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research provides program analysts and Department of Defense (DoD) leadership with an approach to identify problems in real-time for acquisition contracts. Specifically, we develop optimization algorithms to detect unusual changes in acquisition programs' Earned Value data streams. The research is focused on three questions. First, can we predict the contractor provided estimate at complete (EAC)? Second, can we use those predictions to develop an algorithm to determine if a problem will occur in an acquisition program or sub-program? Lastly, can we provide the probability of a problem occurring within a given timeframe? We find three of our models establish statistical significance predicting the EAC. Our four-month model predicts the EAC, on average, within 3.1 percent and our five and six-month models predict the EAC within 3.7 and 4.1 percent. The four-month model proves to present the best predictions for determining the probability of a problem. Our algorithms identify 70% percent of the problems within our dataset, while more than doubling the probability of a problem occurrence compared to current tools in the cost community. Though program managers can use this information to aid analysis, the information we provide should serve as a tool and not a replacement for in-depth analysis of their programs

15. SUBJECT TERMS
Cost, Problem, Algorithm, Simplex, Earned Value Management (EVM)

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Edward White (AFIT/ENC)
U	U	U	UU	133	**19b. TELEPHONE NUMBER** *(Include area code)* 937-255-3636 ext 4540, Edward.white@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18